Natural Language Engineering (2023), **29**, pp. 769–793
doi:10.1017/S1351324921000292

CAMBRIDGE UNIVERSITY PRESS

ARTICLE

A semantic parsing pipeline for context-dependent question answering over temporally structured data

Charles Chen, Razvan Bunescu* and Cindy Marling

School of Electrical Engineering and Computer Science, Ohio University, Athens, OH 45701, USA
*Corresponding author. E-mail: razvan.bunescu@uncc.edu

(Received 9 June 2020; revised 13 September 2021; accepted 15 September 2021; first published online 29 October 2021)

Abstract

We propose a new setting for question answering (QA) in which users can query the system using both natural language and direct interactions within a graphical user interface that displays multiple time series associated with an entity of interest. The user interacts with the interface in order to understand the entity's state and behavior, entailing sequences of actions and questions whose answers may depend on previous factual or navigational interactions. We describe a pipeline implementation where spoken questions are first transcribed into text which is then semantically parsed into logical forms that can be used to automatically extract the answer from the underlying database. The speech recognition module is implemented by adapting a pre-trained long short-term memory (LSTM)-based architecture to the user's speech, whereas for the semantic parsing component we introduce an LSTM-based encoder–decoder architecture that models context dependency through copying mechanisms and multiple levels of attention over inputs and previous outputs. When evaluated separately, with and without data augmentation, both models are shown to substantially outperform several strong baselines. Furthermore, the full pipeline evaluation shows only a small degradation in semantic parsing accuracy, demonstrating that the semantic parser is robust to mistakes in the speech recognition output. The new QA paradigm proposed in this paper has the potential to improve the presentation and navigation of the large amounts of sensor data and life events that are generated in many areas of medicine.

Keywords: Semantic Parsing; Question Answering

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 accumulate rapidly and lead to a substantial data overload for the health provider.

To help doctors more easily browse the wealth of generated patient data, we developed PhysioGraph, a graphical user interface (GUI) that displays various time series of measurements acquired from a patient. As shown in Figure 1, PhysioGraph displays the data corresponding to one day, whereas buttons allow the user to move to the next or previous day. Clicking on any

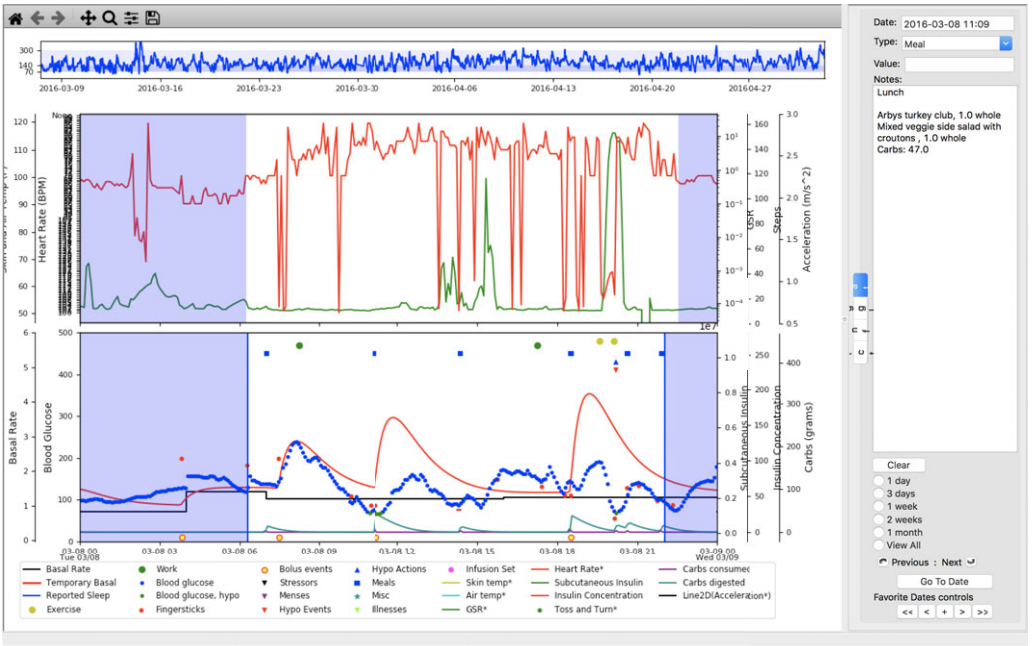

Figure 1. PhysioGraph window displaying one day's worth of patient data (default view). The blue graph at the top shows the entire timeline of blood glucose measurements, over approximately 8 weeks. The top pane below it shows physiological parameters, including heart rate (red) and skin conductivity (green). The bottom pane shows time series of blood glucose (blue), basal insulin (black), and estimated active insulin (red). Discrete life events such as meals and exercise are shown at the top of the bottom pane, whereas boluses are shown at the bottom. When an event is clicked, details are displayed in the pane on the right.

measurement or event displays additional data, such as amount of carbohydrates for a meal or the amount of insulin for a bolus event. The user also has the option of toggling off the display of specific time series, in order to reduce clutter.

While PhysioGraph 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 doctors often found it easier to express their queries in NL, for example, "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, for example, 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 required browsing the entire patient history in the worst case, which was very inefficient. This motivated us to work on a new question answering (QA) paradigm that allows users to express their information needs using NL queries and direct actions within a GUI. To this end, we propose a pipeline architecture comprised of two major modules: first, a speech recognizer transcribes the doctor's spoken question into text; then the text is converted by a semantic parser into a logical form (LF). The LF can be run by an inference engine to automatically retrieve the answer from the database storing the patient data. The new QA paradigm described in this paper has the potential for applications in many areas of medicine where sensor data and life events are

https://doi.org/10.1017/S1351324921000292 Published online by Cambridge University Press

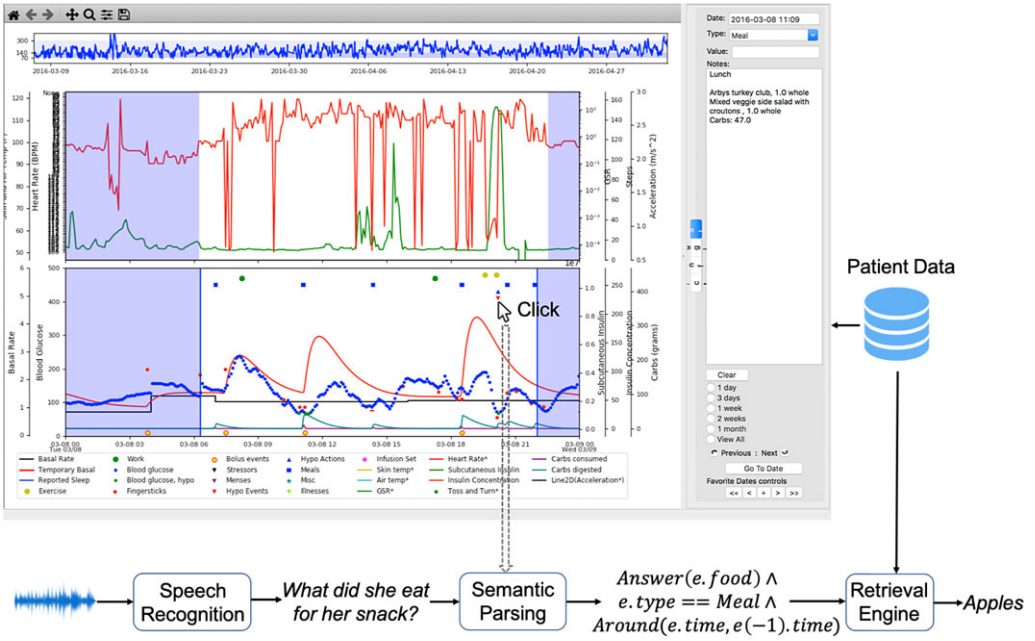

Figure 2. The proposed semantic parsing pipeline for context-dependent question answering.

pervasive. The proposed intelligent user interface will also benefit the exploration and interpretation of data in other domains such as experimental physics, where large amounts of time series data are generated from high-throughput experiments.

The structure of the paper is as follows: Section 2 describes the new QA setting and its major distinguishing features; Section 3 details the speech recognition module and its evaluation; Section 4 focuses on the semantic parsing system and its performance when using correct textual input; Section 5 presents an empirical evaluation of the entire semantic parsing pipeline. The paper ends with related work and concluding remarks.

2. Task definition and distinguishing features

Figure 2 shows an example of interaction with PhysioGraph, where the doctor first clicks on a hypoglycemia event, represented by the red triangle in the bottom pane. To determine what kind of food the patient ate in response to the hypoglycemia event, the doctor then asks the question " *What did she eat for her snack*?". The question is first transcribed into text by the speech recognition module. The transcribed question is then used as input to the semantic parsing system, which produces its formal representation, that is, its LF $Answer(e.food) \wedge e.type = Meal \wedge Around(e.time, e(-1).time)$. Note that the patient had multiple meals (shown as blue squares) on the particular day shown in PhysioGraph. To determine the correct meal intended by the doctor, the semantic parsing module takes into account the fact that the doctor previously clicked on the hypoglycemia event. This serves as an anchoring action and therefore the correct meal should be the one that is close to the hypoglycemia event, as reflected by the clause $Around(e.time, e(-1).time)$ in the LF. In this example, the context dependency is realized by *referencing* the focus event of the previous interaction, formally represented as $e(-1).time$. Once the LF is generated, an answer retrieval engine can easily return the answer by performing a query against the back-end database.

https://doi.org/10.1017/S1351324921000292 Published online by Cambridge University Press

Table 1. Examples of interactions and logical forms

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?
$Answer(e) \wedge Behavior(e_1.value, \text{Up}) \wedge Around(e.time, e_1.time) \wedge e.type = \text{DiscreteType} \wedge e_1.type = \text{HeartRate}$ $\wedge e_1.time = MidAfternoon()$
Q_2: What time did that start?
$Answer(e(-1).time)$
Example 2
Click on Bolus at 8:03pm.
$Click(e) \wedge e.type = \text{Bolus} \wedge e.time = \text{8:03pm}$
Q_3: What did she eat for her snack?
$Answer(e.food) \wedge e.type = \text{Meal} \wedge Around(e.time, e(-1).time)$
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?
$Answer(Any(d.type = \text{Bolus} \wedge Before(d.time, e(-1).time)))$
Example 4
Q_6: What is the first day they have heart rate reported?
$Answer(e.date) \wedge Order(e, 1, Sequence(d, d.type = \text{HeartRate}))$
Example 5
Q_7: Is there another day he goes low in the morning?
$Answer(Any(Hypo(d) \wedge x! = \text{CurrentDate} \wedge x.type = \text{Date} \wedge d.time = Morning(x))$

Table 1 shows additional sample inputs paired with their LFs. For each input, the examples also show relevant previous inputs from the interaction sequence. The LFs often contain constants, such as dates or specific times during the day, which are out of the LF vocabulary. When translating NL queries, these out-of-vocabulary (OOV) constants will be copied from the NL input into the LF using a copy mechanism, as will be described in Section 4.3. A separate copying mechanism will also be used for referencing events in the previous LF.

https://doi.org/10.1017/S1351324921000292 Published online by Cambridge University Press

In the following, we describe a number of major features that, on their own or through their combination, distinguish the proposed QA paradigm from other QA tasks powered by semantic parsing.

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 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 versus NL questions

The user can interact with the system (1) directly within the GUI (e.g. mouse clicks); (2) through NL 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 NL 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 shown at the bottom pane in PhysioGraph, 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.

2.3 Factual queries versus 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 contrast, 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, for example, "toggle on heart rate," or navigating to a different day, for example, "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 LF of a question, and implicitly its answer, may depend on the previous interaction with the system. Examples 1–3 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 occur within questions,

https://doi.org/10.1017/S1351324921000292 Published online by Cambridge University Press

as in question 4 for example. Overall, solving coreferential relations will be essential for good performance.

3. Speech recognition

This section describes the speech recognition system, which is the first module in the proposed semantic parsing pipeline. Given a spoken question or command coming from the user, the task is to transcribe it automatically into text. For this, we adapt the neural end-to-end model proposed by Zeyer *et al.* (2018), a publicly available open source system that was trained on LibriSpeech and which at the time of its publication obtained state-of-the-art results, that is, a word error rate (WER) of 3.82% on the *test-clean* subset of LibriSpeech. While this represents a competitive performance on in-domain data, its WER on the speech recorded from the two doctors in our study was much higher, and thus insufficiently accurate for the semantic parsing module downstream. For lack of free access to a better speech recognizer, our solution was to fine-tune the LibriSpeech-trained model of Zeyer *et al.* (2018) on a small parallel corpus obtained from each doctor, taking advantage of the fact that the encoder–decoder architecture used by this system is relatively simple and straightforward to retrain or fine-tune. First, each input audio file is converted into a Mel-frequency cepstral coefficients (MFCC) representation (Sigurdsson, Petersen, and Lehn-Schiøler 2006). The encoder consists of six layers of bi-directional long short-term memories (LSTMs) that project the MFCC vectors into a sequence of latent representations which are then used to initialize the decoder, a one-layer bi-directional LSTM. The decoder operates on subword-level units, such as characters and graphemes, which are created through byte-pair encoding (BPE) (Sennrich, Haddow, and Birch 2016). An individual grapheme may or may not carry meaning by itself, and may or may not correspond to a single phoneme of the spoken language. The advantage for using subword-level units as output is that it allows recognition of unseen words and does not require a large softmax output which is computationally expensive. Therefore, during training, the manually transcribed text is first processed via BPE to create subword units, which are then used as the actual output targets of the decoder. At each step, the decoder computes attention weights over the sequence of hidden states produced by the encoder and uses them to calculate a corresponding context vector. Together with the current state in the decoder, the context vector is used as input to the softmax that generates the BPE unit for the current step. Once the decoding is complete, all the generated BPE units are merged into a sequence of words. Stochastic gradient descent with global gradient clipping is utilized to train the entire encoder-attention-decoder architecture.

3.1 Experimental evaluation

The speech from two doctors[1] was recorded while they were using PhysioGraph, resulting in two datasets, *Amber* and *Frank*. The corresponding transcripts of the speech, including queries and commands, are obtained by manual annotation. The end-to-end speech recognition model (Zeyer *et al.* 2018), which was originally trained on LibriSpeech, is fine-tuned separately for each of the two datasets. In order to improve the recognition performance, in a second set of experiments, we also apply data augmentation. A more detailed description of the evaluation procedure follows.

The speech recognition system was first trained and tested on the *Amber* dataset using a 10-fold evaluation scenario where the dataset is partitioned into 10 equal size folds, and each fold is used as test data while the remaining 9 folds are used for training. WER is used as the evaluation metric, and the final performance is computed as the average WER over the 10 folds. We denote this system as SR.*Amber* (*Amber* without data augmentation). In addition, we evaluate the speech recognition system with data augmentation. The system in this evaluation scenario is denoted as SR.*Amber* + *Frank*, which means that *Amber* is the target data, while *Frank* is the external data

[1] Amber Healy, DO, and Frank Schwartz, MD, physicians in Ohio University Heritage College of Osteopathic Medicine.

https://doi.org/10.1017/S1351324921000292 Published online by Cambridge University Press

Table 2. Performance of speech recognition system on dataset *Amber* and dataset *Frank* with and without data augmentation. SR.*Amber* and SR.*Frank* are two systems without data augmentation while SR.*Amber* + *Frank* and SR.*Frank* + *Amber* are two systems enhanced with data augmentation. WER (%) is the evaluation metric

	WER (%)	Standard deviation
SR.*Amber*	14.55	3.62
SR.*Amber* + *Frank*	13.90	3.69
SR.*Frank*	15.39	4.23
SR.*Frank* + *Amber*	12.20	3.67

used for data augmentation. We use the same 10 folds of data used in the SR.*Amber* evaluation, where for each of the 10 folds used as test data, all examples in the *Frank* dataset are added to the remaining 9 folds of the *Amber* dataset that are used for training. The final performance of SR.*Amber* + *Frank* is computed as the average WER over the 10 folds.

The WER performance of the speech recognition system on the *Amber* dataset, with and without data augmentation, is shown in the top half of Table 2. The experimental results indicate that SR.*Amber* + *Frank* outperforms SR.*Amber*, which means that data augmentation is able to improve the performance in this evaluation scenario. In order to further investigate the impact of data augmentation, we run a similar evaluation setup on the *Frank* dataset, using the *Amber* dataset for augmentation. The corresponding results for SR.*Frank* and SR.*Frank* + *Amber* are shown in the bottom half of Table 2. Here, too, data augmentation is shown to be beneficial. The improvement of SR.*Frank* + *Amber* over SR.*Frank* and SR.*Amber* + *Frank* over SR.*Amber* is both statistically significant at $p = 0.02$ in a one-tailed paired t-test.

Table 3 reports outputs from the speech recognition system, demonstrating that data augmentation is able to correct some of the mistakes made by the baseline systems.

4. Context-dependent semantic parsing

The role of the semantic parsing module is to take the text version of the doctor's queries and commands and convert it into a formal representation, that is, the *LF*. We first introduce the evaluation datasets in Section 4.1, followed by a description of the neural network architectures in Sections 4.2 and 4.3, and the experimental evaluation in Section 4.4.

4.1 Semantic parsing datasets

To train and evaluate semantic parsing approaches, we created three datasets of sequential interactions: two datasets of real interactions (Section 4.1.1) and a much larger dataset of artificial interactions (Section 4.1.2).

4.1.1 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. The interactions were acquired from two physicians, Frank Schwartz, MD and Amber Healy, DO, and were stored in two separate datasets called *Frank* and *Amber*, respectively. 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 PhysioGraph and

https://doi.org/10.1017/S1351324921000292 Published online by Cambridge University Press

Table 3. Examples of transcriptions generated by SR systems trained on SR.*Frank* and SR.*Amber*, and their augmented versions SR.*Frank* + *Amber* and SR.*Amber* + *Frank*, respectively. *True* refers to the correct transcription

True & SR.*Amber* & SR.*Amber* + *Frank*:
It looks like it happened again on the fifth
What did his sugar do going into the night
True & SR.*Amber* + *Frank*:
And he bolused for it
SR.*Amber*:
Any bolus for it
True & SR.*Amber* + *Frank*:
But he still went high after breakfast
SR.*Amber*:
What he still went high after breakfast
True & SR.*Frank* & SR.*Frank* + *Amber*:
I see that she went low in the middle of the day
I wonder what kind of exercise she's doing this day
True & SR.*Frank* + *Amber*:
How many carbs did the patient consume at that meal
SR.*Frank*:
How many carbs to the patient consume at that meal

clicking on the corresponding events. After each session, the question segments were extracted manually from the speech recordings, transcribed, and timestamped. The transcribed speech was used to fine-tune the speech recognition models, as described in Section 3. 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 LFs, whereas questions were parsed into LFs manually, to be used for training and evaluating the semantic parsing algorithms. Sample interactions and their LFs are shown in Table 1.

A snapshot of the vocabulary for LFs is shown in Table 4, detailing the Event Types, Constants, Functions, Predicates, and Commands. Every life event or physiological measurement stored in the database is represented in the LFs 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 ith previous 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, for example, $e = e(-1)$.

Overall, the LFs in the two datasets have the following statistics:

- The *Frank* dataset contains LFs for 237 interactions, corresponding to 74 mouse clicks and 163 NL queries.
  - The LFs for 43 NL queries *reference* an event from the previous LF.
  - The LFs for 26 NL queries contain *OOV* tokens that can be copied from the NL input.

https://doi.org/10.1017/S1351324921000292 Published online by Cambridge University Press

Table 4. Vocabulary for logical forms

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*, . . .

- The *Amber* dataset contains LFs for 504 interactions, corresponding to 330 mouse clicks and 174 NL queries.
  - The LFs for 97 NL queries *reference* an event from the previous LF.
  - The LFs for 35 NL queries contain *OOV* tokens that can be copied from the NL input.

4.1.2 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 developed an artificial data generator, previously described in Chen *et al.* (2019). To simulate user–GUI interactions, the artificial data

https://doi.org/10.1017/S1351324921000292 Published online by Cambridge University Press

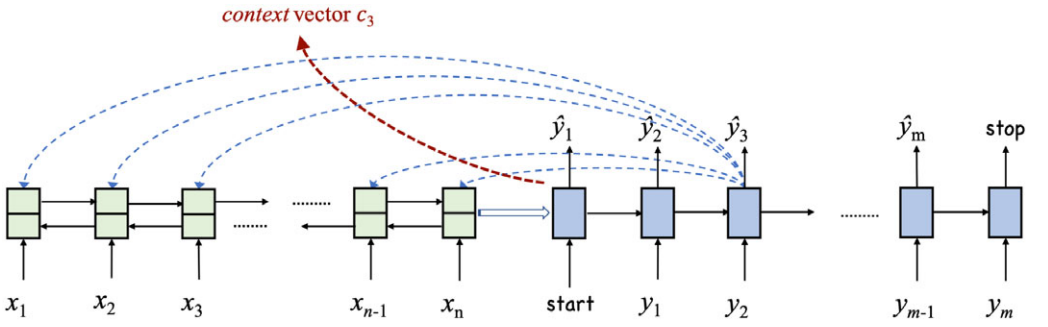

Figure 3. The *SeqGen* model takes a sequence of natural language (NL) tokens as input $X = x_1, \ldots, x_n$ and encodes it with a Bi-LSTM (left, green). The two final states are used to initialize the decoder LSTM (right, blue) which generates the LF sequence $\hat{Y} = \hat{y}_1, \ldots, \hat{y}_m$. The attention-augmented *SeqGen+Att2In* model computes attention weights (blue arrows) and a context vector (red arrow) for each position in the decoder.

generator uses sentence *templates* in order to maintain high-quality sentence structure and grammar. This approach is similar to Weston *et al.* (2016), with the difference that we need a much higher degree of variation such that the machine learning model does not memorize all possible sentences, which resulted in a much richer template database. The *template language* is defined using a recursive grammar, which allows creating as many templates and generating 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 1000 interactions and their LFs: 312 mouse clicks and 688 NL queries.

4.2 Baseline models for semantic parsing

In this section, we describe two baseline models: a standard LSTM encoder–decoder for sequence generation *SeqGen* and its attention-augmented version *SeqGen+Att2In*. The attention-based model will be used later in Section 4.3 as a component in the context-dependent semantic parsing architecture.

As shown in Figure 3, the sequence generation model *SeqGen* uses LSTM (Hochreiter and Schmidhuber 1997) units in an encoder–decoder architecture (Cho *et al.* 2014; Bahdanau, Cho, and Bengio 2015), composed of a bi-directional LSTM for the encoder and a unidirectional LSTM for the decoder. The Bi-LSTM encoder is run over the input sequence X in both directions and the two final states (one from each direction) are concatenated to produce the initial state $\boldsymbol{s}_0$ for the decoder. Starting from the initial hidden state $\boldsymbol{s}_0$, the decoder produces a sequence of states $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_m$. Both the encoder and the decoder use a learned embedding function e for their input tokens. We use $X = x_1, \ldots, x_n$ to represent the sequence of tokens in the NL input, and $Y = y_1, \ldots, y_m$ to represent the tokens in the corresponding LF. We use $Y_t = y_1, \ldots, y_t$ to denote the LF tokens up to position t, and $\hat{Y}$ to denote the entire LF generated by the decoder. A softmax is used by the decoder to compute token probabilities at each position as follows:

$$p(y_t | Y_{t-1}, X) = softmax(\boldsymbol{W}_h \boldsymbol{s}_t) \tag{1}$$

$$\boldsymbol{s}_t = h(\boldsymbol{s}_{t-1}, e(y_{t-1}))$$

The transition function h is implemented by the LSTM unit (Hochreiter and Schmidhuber 1997).

https://doi.org/10.1017/S1351324921000292 Published online by Cambridge University Press

Additionally, the *SeqGen+Att2In* model uses an attention mechanism *Att2In* to compute attention weights and a context vector $\boldsymbol{c}_t$ for each position in the decoder, as follows:

$$e_{tj} = \boldsymbol{v}_a^T \tanh\left(\boldsymbol{W}_a \boldsymbol{f}_j + \boldsymbol{U}_a \boldsymbol{s}_{t-1}\right) \tag{2}$$

$$\alpha_{tj} = \frac{\exp\left(e_{tj}\right)}{\sum_{k=1}^{m} \exp\left(e_{tk}\right)}, \quad \boldsymbol{d}_t = \boldsymbol{c}_t = \sum_{j=1}^{n} \alpha_{tj} \boldsymbol{f}_j$$

We follow the attention mechanism formulation of Bahdanau *et al.* (2015) where, at each time step t in the decoder, the softmax output depends not only on the current LSTM state $\boldsymbol{s}_t$ but also an an additional *context* vector $\boldsymbol{c}_t$ that aims to capture relevant information from the input sequence. The context vector is formulated as a weighted combination of the hidden states $\boldsymbol{f}_j$ computed by the Bi-LSTM encoder over the input NL sequence, using a vector of *attention* weights α_{tj}, one attention weight for each position j in the input sequence. The attention vector is obtained by applying softmax to a vector of unnormalized weights e_{tj} computed from the hidden state $\boldsymbol{f}_j$ in the encoder and the previous decoder state $\boldsymbol{s}_{t-1}$. The context vector $\boldsymbol{d}_t = \boldsymbol{c}_t$ and the current decoder state $\boldsymbol{s}_t$ are then used as inputs to a softmax that computes the distribution for the next token $\hat{y}_t$ in the LF:

$$\hat{y}_t \sim \text{softmax}(\boldsymbol{W}_h \boldsymbol{s}_t + \boldsymbol{W}_d \boldsymbol{d}_t) \tag{3}$$

4.3 Semantic parsing with multiple levels of attention and copying mechanism

Figure 4 shows the proposed context-dependent 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 is composed of three models: *Att2HisIn* attending to the previous input, *Att2HisLF* attending to the previous LF, and the *Att2In* introduced in Section 4.2 that attends to the current input. The copying mechanism is composed of two models: one for handling unseen tokens and one for handling coreference to events in the current and previous LFs.

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

$$\hat{e}_{tk} = \boldsymbol{v}_b^T \tanh\left(\boldsymbol{W}_b \boldsymbol{r}_k + \boldsymbol{U}_b \boldsymbol{s}_{t-1}\right) \tag{4}$$

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

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:

$$\tilde{e}_{tj} = \boldsymbol{v}_c{}^T \tanh\left(\boldsymbol{W}_c \boldsymbol{l}_j + \boldsymbol{U}_c \boldsymbol{s}_{t-1}\right) \tag{5}$$

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

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

The context vector $\boldsymbol{d}_t$ is comprised of the context vectors from the three attention models *Att2In*, *Att2HisIn,* and *Att2HisLF*, and will be used in the decoder softmax as shown previously in

https://doi.org/10.1017/S1351324921000292 Published online by Cambridge University Press

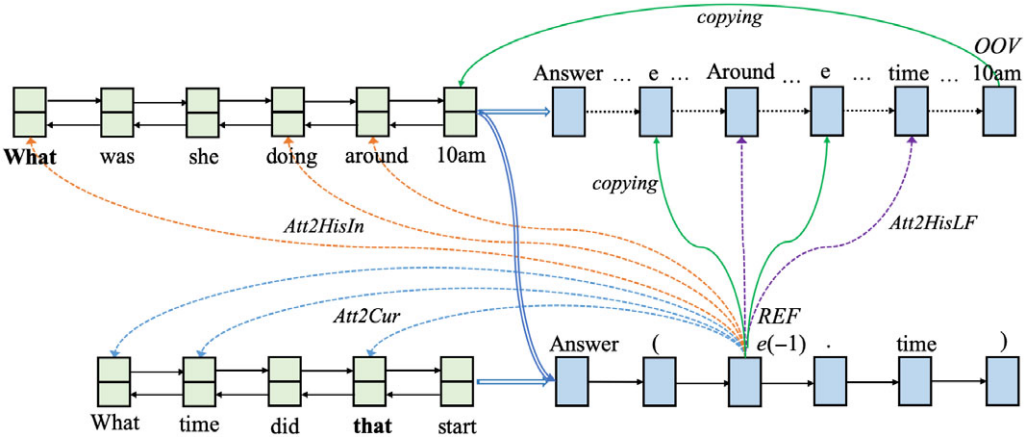

Figure 4. Context-dependent semantic parsing architecture. We use a Bi-LSTM (left) to encode the input and a LSTM (right) as the decoder. Top shows the previous interaction and bottom shows the current interaction. The complete previous LF is Y^{-1} = [*Answer, (, e,), ∧, Around, (, e,., time, OOV,), ∧, e,., type, =, DiscreteType*]. The token *10am* is copied from the input to replace the generated *OOV* token (green arrow). The complete current LF is Y = [*Answer, (, REF,., time,)*]. The entity token *e(-1)* is copied from the previous LF to replace the generated *REF* token (green arrow). To avoid clutter, only a subset of the attention lines (dotted) are shown.

Equation (3):

$$\boldsymbol{d}_t = concat(\boldsymbol{c}_t, \hat{\boldsymbol{c}}_t, \tilde{\boldsymbol{c}}_t) \tag{6}$$

Copying Mechanisms In order to handle OOV tokens and coreference (REF) between entities in the current and the previous LFs, we add two special tokens *OOV* and *REF* to the LF 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:

$$L_{oov}(Y) = -\sum_{j=1}^{X.l}\sum_{t=1}^{Y.l} \log p_o(O_{jt}|\boldsymbol{s}_j^X, \boldsymbol{s}_t^Y) \tag{7}$$

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

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

$$L_{ref}(Y) = -\sum_{j=1}^{\hat{Y}^{-1}.l}\sum_{t=1}^{Y.l} \log p_r(R_{jt}|\boldsymbol{s}_j^{\hat{Y}^{-1}}, \boldsymbol{s}_t^Y) \tag{8}$$

where $\hat{Y}^{-1}.l$ is the length of the previous generated LF, $Y.l$ is the length of the current LF, $\boldsymbol{s}_j^{\hat{Y}^{-1}}$ 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. $R_{jt} \in \{0, 1\}$ is a label that is 1 iff $\hat{y}_j$ is an entity referred by y_t in the next LF Y. We use logistic regression to compute the coreference probability, that is, $p_r(R_{jt} = 1|\boldsymbol{s}_j^{\hat{Y}^{-1}}, \boldsymbol{s}_t^Y) = \sigma(\boldsymbol{w}_r^T[\boldsymbol{s}_j^{\hat{Y}^{-1}}, \boldsymbol{s}_t^Y])$.

https://doi.org/10.1017/S1351324921000292 Published online by Cambridge University Press

4.3.1 Token-level supervised learning: SPAAC-MLE

We use *teacher forcing* (Williams and Zipser 1989) to train the three models to learn which token in the vocabulary (including special tokens *OOV* and *REF*) should be generated. Correspondingly, the two baselines will minimize the following token generation loss:

$$L_{gen}(Y) = -\sum_{t=1}^{Y.l} \log p(y_t | Y_{t-1}, X) \tag{9}$$

where $Y.l$ is the length of the current LF. The supervised learning model *SPAAC-MLE* is obtained by training the semantic parsing architecture from Figure 4 to minimize the sum of the three negative log-likelihood losses:

$$L_{MLE}(Y) = L_{gen}(Y) + L_{oov}(Y) + L_{ref}(Y) \tag{10}$$

At inference time, the decoder is run in *auto-regressive* mode, which means that the input at step t is the previously generated token $\hat{y}_{t-1}$. To alleviate the suboptimality of this greedy procedure, *beam search* is used to generate the LF sequence (Ranzato *et al.* 2016; Wiseman and Rush 2016). During inference, if the generated token at position t is *OOV*, the token from the current input X that has the maximum OOV probability is copied, that is, $\arg\max_{1 \le j \le X.l} p_o(O_j = 1 | \boldsymbol{s}_j^X, \boldsymbol{s}_t^Y)$. Similarly, if the generated entity token at position t is *REF*, the entity token from the previously generated LF sequence $\hat{Y}^{-1}$ that has the maximum coreference probability is copied, that is, $\arg\max_{1 \le j \le Y^{-1}.l} p_r(R_j = 1 | \boldsymbol{s}_j^{\hat{Y}^{-1}}, \boldsymbol{s}_t^{\hat{Y}})$.

4.3.2 Sequence-level reinforcement learning: SPAAC-RL

All models described in this paper are evaluated using sequence-level accuracy, a discrete metric where a generated LF is considered to be correct if it is equivalent with the ground truth LF. 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 multiple generated sequences that are correct, for example, 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 suboptimal performance (Zeng *et al.* 2016; Norouzi *et al.* 2016; Rennie *et al.* 2017; Paulus, Xiong, and Socher 2018). Furthermore, although teacher forcing is widely used for training sequence generation models, it leads to *exposure bias* (Ranzato *et al.* 2016): 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.* (2018), we address these problems 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, that is, $\pi^b(\hat{Y}_{t-1}) = \arg\max_{\hat{y}_t} p(\hat{y}_t | \hat{Y}_{t-1})$; and a sampling policy π^s that samples the action according to the same distribution, that is, $\pi^s(\hat{Y}_{t-1}) \propto p(\hat{y}_t | \hat{Y}_{t-1})$.

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

$$L_{RL} = -R(\hat{Y}^s) \times L_{MLE}(\hat{Y}^s) = -\Big(A(\hat{Y}^s) - A(\hat{Y}^b)\Big) \times L_{MLE}(\hat{Y}^s) \tag{11}$$

https://doi.org/10.1017/S1351324921000292 Published online by Cambridge University Press

Table 5. Sequence-level accuracy on the Artificial dataset and the two Real interactions datasets

Models	Artificial	Frank	Amber
SeqGen	51.8	22.2	18.6
SeqGen+Att2In	72.7	35.4	25.5
SPAAC-MLE	**84.3**	**67.6**	**62.3**
SPAAC-RL	**88.7**	**75.9**	**70.5**

Table 6. Ablation results on the Amber dataset, as we gradually add more components to *SeqGen*

SeqGen	*+ Att2In*	*+ Att2His*	*+ OOVCopy*	*+ REFCopy*
18.6	25.5	45.6	54.4	62.3

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 $\hat{Y}^b$.

4.4 Experimental evaluation

All models are implemented in Tensorflow using dropout to alleviate overfitting. 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 and the LSTM hidden states had dimensionality of 64 and were initialized at random. Optimization is performed with the Adam algorithm (Kingma and Ba 2015), using an initial learning rate of 0.0001 and a minibatch size of 128. All experiments are performed on a single NVIDIA GTX1080 GPU.

For each dataset, we use 10-fold evaluation, where the data are partitioned into 10 folds, one fold is used for testing and the remaining for training. The process is repeated 10 times to obtain test results on all folds. The embeddings for both the input vocabulary and output vocabulary are initialized at random and trained together with the other model parameters. Preliminary experiments did not show a significant benefit from using pre-trained input embeddings, likely due to the limited vocabulary. To model tokens that have not been seen at training time, we train a special $<unknown>$ embedding by assigning it to tokens that appear only once during training. Subword embeddings may provide a better approach to dealing with unknown input words at test time, especially when coupled with contextualized embeddings provided by transformer-based models such as BERT (Devlin *et al.* 2019). We leave these enhancements for future work.

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

The sequence-level accuracy results are reported in Table 5 for the artificial and real datasets. The results demonstrate the importance of modeling context dependency, as the two *SPAAC* models outperform the baselines on all 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. To determine the impact of each model component, in Table 6 we show ablation results on the Amber dataset, as we gradually added more components to the MLE-trained SeqGen baseline. Going from left to right, we show results after adding attention to current input (*Att2In*), attention to history (*Att2His* =

https://doi.org/10.1017/S1351324921000292 Published online by Cambridge University Press

Table 7. Examples generated by *SPAAC-MLE* and *SPAAC-RL* using real interactions. MLE: logical forms by *SPAAC-MLE*. RL: logical forms by *SPAAC-RL*. True: manually annotated LFs

Well the Finger Stick is 56.
True & MLE & RL:
$e.type = \text{Fingerstick} \wedge e.value = 56$
It looks like she suspended her pump.
True & MLE & RL:
$Suspended(e) \wedge Around(e.time, e(-1).time)$
Let's look at the next day.
True & MLE & RL:
$DoSetDate(\text{CurrentDate} + 1)$
See if he went low.
True & MLE & RL:
$Answer(Any(e, Hypo(e)))$
Let's see what kind of exercise that is, where the steps are high?
True & RL:
$Answer(e.kind) \wedge e.type = \text{Exercise} \wedge Around(e.time, e_1.time) \wedge e_1.type = \textbf{StepCount} \wedge \boldsymbol{High}(\boldsymbol{e_{1.value}})$
MLE:
$Answer(e.kind) \wedge e.type = \text{Exercise} \wedge Around(e.time, e_1.time) \wedge e_1.type = \textbf{Exercise} \wedge \boldsymbol{e_{1.type=}}\textbf{Exercise}$
Click on the exercise.
True & RL:
$\boldsymbol{Doclick}(e) \wedge e.type = \text{Exercise}$
MLE:
$\boldsymbol{Answer}(e) \wedge e.type = \text{Exercise}$

Att2HisIn + *Att2HisLF*), the copy mechanism for OOV tokens (*OOVCopy*), and the copy mechanism for coreference (*REFCopy*). Note that the last result in the table corresponds to *SPAAC-MLE* = *SeqGen* + *Att2In* + *Att2His* + *OOVCopy* + *REFCopy*. The results show substantial improvements after adding each attention component and copying mechanism. The improvements due to the two copy mechanisms are expected, given the significant number of OOV tokens and references that appear in the NL queries. According to the statistics presented in Section 4.1.1, across the two real interactions datasets, 18.1% of LFs for NL queries contain OOV tokens, while 41.5% of LFs for NL queries make references to the previous LF. By definition, the *SeqGen* baseline cannot produce the correct LF for any such query.

Tables 7 and 8 show examples generated by the *SPAAC* models on the *Frank* dataset. Analysis of the generated LFs 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 the sampled LFs $\hat{Y}^s$ and the baseline LFs $\hat{Y}^b$, we found that in some cases the tokens for event types in $\hat{Y}^s$ are identical with the wrong event types created in the baseline LFs $\hat{Y}^b$.

4.4.1 Data augmentation

Inspired by the fact that data augmentation improves the performance of speech recognition (Section 3), we applied data augmentation to the semantic parsing system as well. Specifically, we train and test the semantic parsing system, for both models *SPAAC-MLE* and *SPAAC-RL*, on the

https://doi.org/10.1017/S1351324921000292 Published online by Cambridge University Press

Table 8. Examples generated by *SPAAC-MLE* and *SPAAC-RL* using artificial interactions. MLE: logical forms generated by *SPAAC-MLE*. RL: logical forms generated by *SPAAC-RL*. True: manually annotated logical forms

Does he always get some sleep around 4:30pm?
True & MLE & RL:
$Answer(Cond(Around(x, 4{:}30pm) => Any(e.type = \text{ReportedSleep} \wedge e.time = x)))$
Is it the first week of the patient?
True & MLE & RL:
$Answer(Week(\text{CurrentDate}) = x) \wedge Order(x, 1, Sequence(e, e.type = \text{Week}))$
Does she ever get some rest around 5:37pm?
True & MLE & RL:
$Answer(Any(e.type = \text{ReportedSleep} \wedge Around(e.time, 5{:}37pm)))$
When is the first time he changes his infusion set?
True & MLE & RL:
$Answer(e.date) \wedge Order(e, 1, Sequence(e, e.type = \text{InfusionSet}))$
How many months she has multiple exercises?
True & RL:
$Answer(Count(x, Count(e, e.type = \text{Exercise} \wedge e.date = x) > 1 \wedge x.type = \textbf{Month}))$
MLE:
$Answer(Count(x, Count(e, e.type = \text{Exercise} \wedge e.date = x) > 1 \wedge x.type = \textbf{Week}))$
Toggle so we can see fingersticks.
True & RL:
$DoToggle(\text{On}, \textbf{FingerSticks})$
MLE:
$DoToggle(\text{On}, \textbf{BGL})$

dataset *Amber* using a 10-fold evaluation scenario similar to the evaluation of the speech recognition system described in Section 3. This system is denoted as SP.*Amber* (Semantic Parsing of *Amber* interactions without data augmentation). The dataset *Amber* is partitioned into 10 equal size folds. For each experiment, nine folds of data are used to train the semantic parsing system and one fold of data is used to evaluate its performance. This process is repeated 10 times, each time using a different fold as test data. The final performance of SP.*Amber* is computed as the average sequence-level accuracy over the 10 folds. In a second set of experiments, we apply data augmentation to SP.*Amber*. The system in this evaluation scenario is denoted as SP.*Amber* + *Frank*, which means that *Amber* is the target data and *Frank* is the external data used for augmentation. In this evaluation scenario, we use the same split in 10 folds as the one used for the SP.*Amber* experiments. In each experiment, all examples in the *Frank* dataset are added to the nine folds of data from *Amber* that are used for training. After training on the nine folds of *Amber* and the data from *Frank*, the remaining one fold of data, which comes from the *Amber* dataset, is used to evaluate the performance of SP.*Amber* + *Frank*. The final performance of SP.*Amber* + *Frank* is computed as the average sequence-level accuracy over the 10 folds. The results on the *Amber* data with and without augmentation are shown in the first section of Table 9. To further evaluate the impact of data augmentation, we also run the symmetric evaluation where *Frank* is the target data and *Amber* is used as external data. The corresponding results are shown in the second section of Table 9. Finally, we evaluate the semantic parsing system on the union of the

https://doi.org/10.1017/S1351324921000292 Published online by Cambridge University Press

Table 9. Sequence-level accuracy (%) of semantic parsing systems on datasets *Amber* and *Frank*. SP.*Amber* and SP.*Frank* are the original systems without data augmentation while SP.*Amber* + *Frank* and SP.*Frank* + *Amber* are the two systems enhanced with data augmentation. SP.*All* is a system trained and tested on the union of the two datasets

	SeqGen	*SeqGen* + *Att2In*	*SPAAC-MLE*	*SPAAC-RL*
SP.*Amber*	18.6	25.5	62.3	70.5
SP.*Amber* + *Frank*	20.2	27.9	66.4	73.0
SP.*Frank*	22.2	35.4	67.6	75.9
SP.*Frank* + *Amber*	24.5	36.3	**68.7**	**76.3**
SP.*All*	20.9	32.9	67.6	73.4

two datasets *Amber* ∪ *Frank*, using the same 10-fold evaluation scenario. This system is denoted as SP.*All*, with its sequence-level accuracy shown in the last section of the table.

Overall, the results in Table 9 show that data augmentation is beneficial for both models *SPAAC-MLE* and *SPAAC-RL*, which continue to do substantially better than the baseline *SeqGen*. When evaluated on the union of the two datasets, the performance of SP.*All* is, as expected, between the accuracy obtained by SP.*Amber* + *Frank* and SP.*Frank* + *Amber*.

Table 10 shows example LFs where data augmentation is able to correct mistakes made by the original SP.*Amber* and SP.*Frank* systems.

5. Semantic parsing pipeline evaluation

The two main modules of the QA pipeline – *speech recognition* and *semantic parsing* – were independently evaluated in Sections 3 and 4, respectively. In this section, we evaluate the performance of the entire semantic parsing pipeline, where the potentially noisy text output by the speech recognizer is used as input for the semantic parsing model. To make results comparable, we used the same 10-fold partition that was used in Sections 3 and 4. The last column in Table 11 shows the sequence-level accuracy of the LFs when running the entire semantic parsing pipeline on the doctors' speech. Because errors made by the speech recognition system are propagated through the semantic parsing module, the pipeline accuracy is lower than the accuracy of standalone semantic parsing shown in the second column of results. However, inspection of the generated LFs revealed cases where errors introduced by the speech recognition system did not have a negative impact on the final LFs. An example is shown in the first section of Table 12, where the speech recognition system mistakenly transcribes *"for"* as *"a"*. However, when run on this slightly erroneous text, the semantic parsing system still generates the correct LF. Another example is shown in the second section, where even though the speech recognition system produces the extra word *"again,"* the semantic parsing system still outputs the correct LF. The fact that semantic parsing is robust to some speech recognition errors helps explain why SP.*Frank* is almost as good as SP.*Frank* + *Amber*, even though the speech recognizer for SP.*Frank* makes more errors than the one trained for SP.*Frank* + *Amber*. There are, however, cases where the speech recognizer makes multiple errors in the same sentence, as in the example shown in the third section of Table 12, for which the semantic parsing system is unable to recover the correct LF. In this case, instead of returning just one BGL value, corresponding to a particular hyperglycemia event, the system returns all the BGL values recorded that day, which should make it obvious to the user that the system did not understand the question. For this kind of errors, the user has the option to find the correct answer to their query by interacting directly with the GUI using mouse clicks. There are also more subtle errors where the user receives a partially overlapping answer, as shown in the second example

https://doi.org/10.1017/S1351324921000292 Published online by Cambridge University Press

Table 10. Logical forms generated by SP.*Amber* + *Frank* versus SP.*Amber*, and SP.*Frank* + *Amber* versus SP.*Frank*. *True* refers to the manually annotated logical forms

How high did she go?
True & SP.*Amber* & SP.*Amber* + *Frank*:
$Answer(e.value) \wedge Lowest(e.value) \wedge e.type = \text{BGL}$
How much insulin did she give herself?
True & SP.*Amber* & SP.*Amber* + *Frank*:
$Answer(e.value) \wedge e.type = \text{Bolus}$
What was her action?
True & SP.*Amber* + *Frank*:
$Answer(e) \wedge \boldsymbol{Around(e.time, e(-1).time)} \wedge e.type = \text{DiscreteType}$
SP.*Amber*:
$Answer(e) \wedge e.type = \textbf{Exercise}$
Did she get too much insulin with breakfast?
True & SP.*Amber* + *Frank*:
$Answer(e.value) \wedge e.type = \textbf{Bolus} \wedge Around(e.time, e_1.time) \wedge e_1.type = \text{Breakfast}$
SP.*Amber*:
$Answer(e.value) \wedge e.type = \text{Breakfast} \wedge Around(e.time, \boldsymbol{e(-1)}.time)$
She's high at night that night.
True & SP.*Frank* & SP.*Frank* + *Amber*:
$High(e.value) \wedge e.type = \text{BGL} \wedge e.time = Night()$
See what she ate.
True & SP.*Frank* & SP.*Frank* + *Amber*:
$Answer(e.food) \wedge e.type = \text{Meal} \wedge Around(e.time, e(-1).time)$
I wonder what happened then.
True & SP.*Frank* + *Amber*:
$Answer(e) \wedge Around(e.time, e(-1).time) \wedge e.type = \text{DiscreteType}$
SP.*Frank*:
$Answer(\boldsymbol{e.food}) \wedge e.type = \textbf{Bolus} \wedge Around(e.time, e(-1).time)$
if you look around 10 o'clock at night, sweat goes up.
True & SP.*Frank* + *Amber*:
$Behavior(e.value, \text{Up}) \wedge Around(e.time, 10) \wedge e.type = \text{GSR}$
SP.*Frank*:
$Behavior(e.value, \textbf{Down}) \wedge Around(e.time, 10) \wedge e.type = \textbf{BGL}$

from Table 10, where instead of returning all types of discrete events anchored around a certain event, the SP.Amber system returns only Exercise events, at anytime during the day. These types of errors show that, in order to become of practical utility, the system could benefit from features that enabled the user to effortlessly determine whether the query was properly understood. Possible features range from computing and displaying confidence values associated with the outputs, to making the LF computed by the system available to the user, to using explainability methods such as SHAP (Lundberg and Lee 2017) that show the inputs that are most responsible for a potentially incorrect LF, thus giving the user the option to reformulate their query or even correct the output in an active learning scenario. While exploring these features is left for future

https://doi.org/10.1017/S1351324921000292 Published online by Cambridge University Press

Table 11. The performance of the entire semantic parsing pipeline. SP.*Amber* and SP.*Frank* are the two systems without data augmentation while SP.*Amber* + *Frank* and SP.*Frank* + *Amber* are the two systems enhanced with data augmentation. Word error rate (WER) and Sequence-level accuracy (%) are the evaluation metrics

	Speech recognition WER(%)	Semantic parsing Acc(%)	Pipeline Acc(%)
SP.*Amber*	14.55	70.5	65.9
SP.*Amber* + *Frank*	13.90	73.0	67.2
SP.*Frank*	15.39	75.9	73.7
SP.*Frank* + *Amber*	**12.20**	**76.3**	**73.8**

Table 12. Transcriptions (SR) and logical forms (SP) generated by the pipeline versus corresponding manually annotations (True text and True LF)

True text:
How much insulin did she give herself **for** correction?
SR.*Amber* + *Frank*:
How much insulin did she give herself **a** correction?
True LF & SP.*Amber* + *Frank*:
$Answer(e.value) \wedge e.type = \text{Bolus}$
True text:
Looks like she actually went low later in the day what time was that?
SR.*Frank* + *Amber*:
Looks like she actually went low **again** later in the day what time was that?
True LF & SP.*Frank* + *Amber*:
$Answer(e.time) \wedge Hypo(e)$
True text:
How **high** did **her** sugar **get**?
SR.*Frank* + *Amber*:
How **I** did sugar **it**?
True LF:
$Answer(e.value) \wedge \mathbf{Highest(\ e.value)} \wedge e.type = \text{BGL} \wedge \mathbf{Around(\ e.time,\ e(-1).time)}$
SP.*Frank* + *Amber*:
$Answer(e.value) \wedge e.type = \text{BGL}$

work, we believe that the system described in this paper represents an important first step toward realizing a new QA paradigm in which users express their information needs relative to a GUI using NL queries either exclusively or in combination with direct GUI interactions through mouse clicks.

6. Related work

Speech Recognition In conventional speech recognition systems, hidden Markov model (HMM)-based approaches and template-based approaches are commonly used (Matarneh *et al.* 2017).

https://doi.org/10.1017/S1351324921000292 Published online by Cambridge University Press

Given a phonetic pronunciation lexicon, the HMM approaches usually operate on the phone level. Handling OOV words is not straightforward and increases the complexity significantly. In recent years, deep learning has been shown to be highly effective in acoustic modeling (Hinton and Salakhutdinov 2006; Bengio 2009; Hinton *et al*. 2012), by employing deep architectures where each layer in the architecture models an increasingly higher-level abstraction of the data. Furthermore, the encoder–decoder framework with neural networks has shown promising results for speech recognition (Chan *et al*. 2016; Doetsch, Zeyer, and Ney 2016; Toshniwal *et al*. 2017; Zeyer *et al*. 2018; Baskar *et al*. 2019; Li *et al*. 2019; Pham *et al*. 2019). When trained end-to-end, neural models operate directly on words, subwords, or characters/graphemes, which removes the need for a pronunciation lexicon and explicit phones modeling, highly simplifying decoding, as in Zeyer *et al*. (2018). Other models, such as inverted HMMs (Doetsch *et al*. 2017) and the recurrent neural aligner (Sak *et al*. 2017), can be interpreted in the same encoder–decoder framework, but often employ some variant of hard latent monotonic attention instead of soft attention used by Zeyer *et al*. (2018).

In our approach, we used fine-tuning to adapt the end-to-end speech recognition of Zeyer *et al*. (2018) to the doctors' speech. This is a straightforward adaptation approach that does not require changing the structure of the adapted module or introducing external modules during inference. Other approaches explore the adaptation of the language model used by the speech recognition system. To accurately estimate the impact of language model adaptation, Mdhaffar *et al*. (2019) first perform a qualitative analysis and then conduct experiments on a dataset in French, showing that language model adaptation reduces the WER of speech recognition significantly. Raju *et al*. (2019) incorporate a language model into a speech recognition system and train it on heterogenous corpora so that personalized bias is reduced. Corona, Thomason, and Mooney (2017) incorporate a language model and a semantic parser into a speech recognition system, obtaining a significant improvement over a state-of-the-art baseline in terms of accuracy for both transcription and semantic understanding. Bai *et al*. (2019) enhance speech recognition by incorporating a language model via a training approach based on knowledge distillation; the recurrent neural network language model, which is trained on a large corpus, computes a set of soft labels to guide the training of the speech recognition system.

Semantic Parsing, QA, and Context Dependency Semantic parsing, which is mapping text in NL to LFs, has emerged as an important component for building QA systems, as in Liang (2016), Jia and Liang (2016), Zhong, Xiong, and Socher (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; Long, Pasupat, and Liang 2016; Iyyer, Yih, and Chang 2017). Long *et al*. (2016) consider the task of learning a context-dependent mapping from NL utterances to denotations in three scenarios: Alchemy, Scene, and Tangrams. Using only denotations at training time, the search space for LFs is much larger than that of the context-dependent utterances. To handle this challenge, the authors perform successive projections of the full model onto simpler models that operate over equivalence classes of LFs. Iyyer *et al*. (2017) explore a semantic parsing task for answering sequences of simple but inter-related questions in a conversational QA setting. A dataset of over 6000 question sequences is collected, where each question sequence inquires about semi-structured tables in Wikipedia. To solve this sequential QA task, a novel dynamic neural semantic parsing model is designed, which is trained using a weakly supervised reward guided search. Although these approaches take into account sequential dependencies between questions or sentences, the setting proposed in this paper has a number of significant distinguishing features, such as the importance of time – data are represented naturally as multiple time series of events – and the anchoring on a GUI that enables direct interactions through mouse clicks as well as combinations of factual queries and interface commands.

https://doi.org/10.1017/S1351324921000292 Published online by Cambridge University Press

Dong and Lapata (2016) use an attention-enhanced encoder–decoder architecture to learn the LFs from NL without using hand-engineered features. Their proposed Seq2Tree architecture is able to capture the hierarchical structure of LFs. Jia and Liang (2016) propose a sequence-to-sequence recurrent neural network model where a novel attention-based copying mechanism is used for generating LFs from questions. The copying mechanism has also been investigated by Gulcehre *et al*. (2016) and Gu *et al*. (2016) in the context of a wide range of NLP applications. More recently, Dong and Lapata (2018) propose a two-stage semantic parsing architecture, where the first step constructs a rough sketch of the input meaning, which is then completed in the second step by filling in details such as variable names and arguments. One advantage of this approach is that the model can better generalize across examples that have the same sketch, that is, basic meaning. This process could also benefit the models introduced in this paper, where constants such as times and dates would be abstracted out during sketch generation. Another idea that could potentially benefit our context-dependent semantic parsing approach is to rerank an n-best list of predicted LFs based on quality-measuring scores provided by a generative reconstruction model or a discriminative matching model, as was recently proposed by Yin and Neubig (2019).

The semantic parsing models above considered sentences in isolation. In contrast, generating correct LFs in our task required modeling sequential dependencies between LFs. In particular, we modeled coreference between events mentioned in different LFs by repurposing the copying mechanism originally used for modeling OOV tokens.

7. Conclusion and future work

We introduced a new QA paradigm in which users can query a system using both NL and direct interactions (mouse clicks) within a GUI. Using medical data acquired from patients with type 1 diabetes as a case study, we proposed a pipeline implementation where a speech recognizer transcribes spoken questions and commands from doctors into a semantically equivalent text that is then semantically parsed into a LF. Once a spoken question has been mapped to its formal representation, its answer can be easily retrieved from the underlying database. The speech recognition module is implemented by adapting a pre-trained LSTM-based architecture to the user's speech, whereas for the semantic parsing component we introduce an LSTM-based encoder–decoder architecture that models context dependency through copying mechanisms and multiple levels of attention over inputs and previous outputs. Correspondingly, we created a dataset of real interactions and a much larger dataset of artificial interactions. When evaluated separately, with and without data augmentation, both models are shown to substantially outperform several strong baselines. Furthermore, the full pipeline evaluation shows only a limited degradation in semantic parsing accuracy, demonstrating that the semantic parser is robust to mistakes in the speech recognition output. The solution proposed in this paper for the new QA paradigm has the potential for applications in many areas of medicine where large amounts of sensor data and life events are pervasive. The approaches introduced in this paper could also benefit other domains, such as experimental physics, where large amounts of time series data are generated from high-throughput experiments.

Besides the ideas explored in Section 5 for increasing the system's practical utility, one avenue for future work is to develop an end-to-end neural model which takes the doctor's spoken questions as input and generates the corresponding LFs directly, instead of producing text as an intermediate representation. Another direction is to jointly train the speech recognition and semantic parsing system, which would enable the speech recognition model to focus on words that are especially important for decoding the correct LF. Finally, expanding the two datasets with more examples of NL interactions is expected to lead to better performance, as well as provide a clearer measure of its performance in a real use scenario.

https://doi.org/10.1017/S1351324921000292 Published online by Cambridge University Press

The two datasets and the implementation of the systems presented in Section 4 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 Amber Healy 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 constructive comments.

Conflicts of interest. The authors declare none.

References

Artzi Y. and Zettlemoyer L. (2011). Bootstrapping semantic parsers from conversations. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, Edinburgh, UK, pp. 421–432.

Bahdanau D., Cho K. and Bengio Y. (2015). Neural machine translation by jointly learning to align and translate. In **Bengio Y. and LeCun Y.**, (eds), *3rd International Conference on Learning Representations, ICLR*, San Dieco, CA.

Bai Y., Yi J., Tao J., Tian Z. and Wen Z. (2019). Learn spelling from teachers: transferring knowledge from language models to sequence-to-sequence speech recognition. In *Interspeech*, Graz, Austria, pp. 3795–3799.

Baskar M.K., Watanabe S., Astudillo R., Hori T., Burget L. and Černocký J. (2019). Semi-supervised sequence-to-sequence ASR using unpaired speech and text. In *Interspeech*, Graz, Austria, pp. 3790–3794.

Bengio Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning* **2**(1), 1–127.

Chan W., Jaitly N., Le Q. and Vinyals O. (2016). Listen, attend and spell: a neural network for large vocabulary conversational speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Shanghai, China, pp. 4960–4964.

Chen C., Mirshekarian S., Bunescu R. and Marling, C. (2019). From physician queries to logical forms for efficient exploration of patient data. In *IEEE International Conference on Semantic Computing (ICSC)*, Newport Beach, CA, pp. 371–374.

Cho K., van Merriënboer B., Gülçehre Ç., Bahdanau D., Bougares F., Schwenk H. and Bengio Y. (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*, Doha, Qatar, pp. 1724–1734.

Corona R., Thomason J. and Mooney R. (2017). Improving black-box speech recognition using semantic parsing. In *Proceedings of the 8th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, Taipei, Taiwan, pp. 122–127.

Devlin J., Chang M.-W., Lee K. and Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota. Association for Computational Linguistics, pp. 4171–4186.

Doetsch P., Hannemann M., Schlüter R. and Ney H. (2017). Inverted alignments for end-to-end automatic speech recognition. *IEEE Journal of Selected Topics in Signal Processing* **11**(8), 1265–1273.

Doetsch P., Zeyer A. and Ney H. (2016). Bidirectional decoder networks for attention-based end-to-end offline handwriting recognition. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, Shenzhen, China, pp. 361–366.

Dong L. and Lapata M. (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)*, Berlin, Germany, pp. 33–43.

Dong L. and Lapata M. (2018). Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia. Association for Computational Linguistics, pp. 731–742.

Gu J., Lu Z., Li H. and Li V.O. (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)*, Berlin, Germany, pp. 1631–1640.

Gulcehre C., Ahn S., Nallapati R., Zhou B. and Bengio Y. (2016). Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany, pp. 140–149.

Harabagiu S., Hickl A., Lehmann J. and Moldovan D. (2005). Experiments with interactive question-answering. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, Ann Arbor, MI, pp. 205–214.

Hinton G., Deng L., Yu D., Dahl G., Mohamed A.-R., Jaitly N., Senior A., Vanhoucke V., Nguyen P., Kingsbury B. and others (2012). Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine* **29**, 82–97.

https://doi.org/10.1017/S1351324921000292 Published online by Cambridge University Press

Hinton G.E. and Salakhutdinov R.R. (2006). Reducing the dimensionality of data with neural networks. *Science* **313**(5786), 504–507.

Hochreiter, S. and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation* **9**(8), 1735–1780.

Iyyer M., Yih W.-T. and Chang M.-W. (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)*, Vancouver, Canada, pp. 1821–1831.

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

Kelly D. and Lin J. (2007). Overview of the TREC 2006 ciQA task. *In ACM SIGIR Forum*, vol. **41**, pp. 107–116.

Kingma D.P. and Ba J. (2015). Adam: a method for stochastic optimization. In *ICLR 2015, the 3rd International Conference on Learning Representations, Conference Track Proceedings*, San Diego, CA.

Li J., Lavrukhin V., Ginsburg B., Leary R., Kuchaiev O., Cohen J. M., Nguyen H. and Gadde R. T. (2019). Jasper: an end-to-end convolutional neural acoustic model. In *Interspeech*, Graz, Austria, pp. 71–75.

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

Long R., Pasupat P. and Liang P. (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)*, Berlin, Germany, pp. 1456–1465.

Lundberg S. M. and Lee S.-I. (2017). A unified approach to interpreting model predictions. *In Advances in Neural Information Processing Systems*, vol. **30**, Long Beach, CA, pp. 1–10.

Matarneh R., Maksymova S., Lyashenko V. and Belova N. (2017). Speech recognition systems: a comparative review. *IOSR Journal of Computer Engineering* **19**(5), 71–79.

Mdhaffar S., Esteve Y., Hernandez N., Laurent A., Dufour R. and Quiniou S. (2019). Qualitative evaluation of ASR adaptation in a lecture context: application to the PASTEL corpus. In *Interspeech*, Graz, Austria, pp. 569–573.

Norouzi M., Bengio S., Jaitly N., Schuster M., Wu Y., Schuurmans D. et al. (2016). Reward augmented maximum likelihood for neural structured prediction. In *Advances in Neural Information Processing Systems 29*, Barcelona, Spain, pp. 1723–1731.

Paulus R., Xiong C. and Socher R. (2018). A deep reinforced model for abstractive summarization. In *The 6th International Conference on Learning Representations (ICLR), Conference Track Proceedings*, Vancouver, Canada.

Pham N.-Q., Nguyen T.-S., Niehues J., Müller M. and Waibel A. (2019). Very deep self-attention networks for end-to-end speech recognition. In *Interspeech*, Graz, Austria, pp. 66–70.

Raju A., Filimonov D., Tiwari G., Lan G. and Rastrow A. (2019). Scalable multi corpora neural language models for ASR. In *Interspeech*, Graz, Austria, pp. 3910–3914.

Ranzato M., Chopra S., Auli M. and Zaremba W. (2016). Sequence level training with recurrent neural networks. In *The 4th International Conference on Learning Representations (ICLR), Conference Track Proceedings*, San Juan, Puerto Rico.

Rennie S.J., Marcheret E., Mroueh Y., Ross J. and Goel V. (2017). Self-critical sequence training for image captioning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, Hawaii pp. 7008–7024.

Sak H., Shannon M., Rao K. and Beaufays F. (2017). Recurrent neural aligner: an encoder-decoder neural network model for sequence to sequence mapping. In *Interspeech*, Stockholm, Sweden, pp. 1298–1302.

Sennrich R., Haddow B. and Birch A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany, pp. 1715–1725.

Sigurdsson S., Petersen K.B. and Lehn-Schiøler T. (2006). Mel frequency cepstral coefficients: an evaluation of robustness of MP3 encoded music. In *International Conference on Music Information Retrieval (ISMIR)*, Victoria, Canada, pp. 286–289.

Toshniwal S., Tang H., Lu L. and Livescu K. (2017). Multitask learning with low-level auxiliary tasks for encoder-decoder based speech recognition. In *Interspeech*, Stockholm, Sweden, pp. 3532–3536.

Weston J., Bordes A., Chopra S. and Mikolov T. (2016). Towards AI-complete question answering: a set of prerequisite toy tasks. In *The 4th International Conference on Learning Representations (ICLR), Conference Track Proceedings*, San Juan, Puerto Rico.

Williams R. J. and Zipser D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation* **1**(2), 270–280.

Wiseman S. and Rush A.M. (2016). Sequence-to-sequence learning as beam-search optimization. In *Empirical Methods in Natural Language Processing (EMNLP)*, Austin, TX, pp. 1296–1306.

Yin P. and Neubig G. (2019). Reranking for neural semantic parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy. Association for Computational Linguistics, pp. 4553–4559.

Zeng W., Luo W., Fidler S. and Urtasun R. (2016). Efficient summarization with read-again and copy mechanism. *CoRR*, abs/1611.03382.

Zettlemoyer L.S. and Collins M. (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*, Singapore, pp. 976–984.

https://doi.org/10.1017/S1351324921000292 Published online by Cambridge University Press

Zeyer A., Irie K., Schlüter R. and Ney H. (2018). Improved training of end-to-end attention models for speech recognition. In *Interspeech*, Hyderabad, India, pp. 7–11.

Zhong V., Xiong C. and Socher R. (2017). Seq2SQL: generating structured queries from natural language using reinforcement learning. CoRR, abs/1709.00103.

A. Artificial Data Generator

An artificial data generator was designed and implemented to simulate doctor–system interactions, with sentence *templates* defining the skeleton of each entry in order to maintain high-quality sentence structure and grammar. A context free grammar is used to implement a *template*

Table A1. Examples of generation of artificial samples

week_days → Monday | Tuesday | . . . | Sunday
daily_intervals → Morning | Afternoon | Evening | Night
daily_intervals_logic → *Morning* | *Afternoon* | *Evening* | *Night*
any_event → HeartRate | Bolus | BGL
any_event_logic → *HeartRate* | *Bolus* | *BloodGlucoseLevel*
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))

https://doi.org/10.1017/S1351324921000292 Published online by Cambridge University Press

language that can specify a virtually unlimited number of templates and generate as many examples as desired. 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?".

The implementation allows for the definition of any number of non-terminals, which are called *types*, and any number of *templates*, which are the possible right-hand sides of the starting symbol *S*. The doctor–system interactions can be categorized into three types: *questions*, *statements*, and *clicks*, where templates can be defined for each type, as shown in Table A1. Given the set of types and templates, a virtually unlimited number of sentences can be derived to form the artificial dataset. Since the sentence generator chooses each template randomly, the order of sentences in the output dataset will be random. However, two important aspects of the real interactions are context dependency and coreference. To achieve context dependency, the implementation allows for more complex *combo templates* where multiple templates are forced to come in a predefined order. It is also possible to specify groups of templates, via tagging, and 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 A1 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].

Cite this article: Chen C, Bunescu R and Marling C (2023). A semantic parsing pipeline for context-dependent question answering over temporally structured data. *Natural Language Engineering* **29**, 769–793. https://doi.org/10.1017/S1351324921000292