Knowledge-Aware Conversational Semantic Parsing Over Web Tables

Yibo Sun[1], Duyu Tang[2], Jingjing Xu[3], Nan Duan[2], Xiaocheng Feng[1], Bing Qin[1], Ting Liu[1], and Ming Zhou[2]

[1] Harbin Institute of Technology, Harbin, China
[2] Microsoft Research Asia, Beijing, China
[3] MOE Key Lab of Computational Linguistics, School of EECS, Peking University
{ybsun,xcfeng,qinb,tliu}@ir.hit.edu.cn
{dutang,nanduan,mingzhou}@microsoft.com
{jingjingxu}@pku.edu.cn

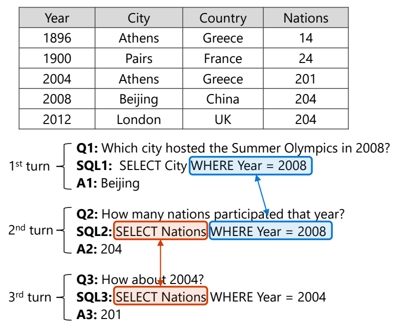
Abstract. Conversational semantic parsing over tables requires knowledge acquiring and reasoning abilities, which have not been well explored by current state-of-the-art approaches. Motivated by this fact, we propose a knowledge-aware semantic parser to improve parsing performance by integrating various types of knowledge. In this paper, we consider three types of knowledge, including grammar knowledge, expert knowledge, and external resource knowledge. First, grammar knowledge empowers the model to effectively replicate previously generated logical form, which effectively handles the co-reference and ellipsis phenomena in conversation Second, based on expert knowledge, we propose a decomposable model, which is more controllable compared with traditional end-to-end models that put all the burdens of learning on trial-and-error in an end-to-end way. Third, external resource knowledge, i.e., provided by a pre-trained language model or an entity typing model, is used to improve the representation of question and table for a better semantic understanding. We conduct experiments on the SequentialQA dataset. Results show that our knowledge-aware model outperforms the state-of-the-art approaches. Incremental experimental results also prove the usefulness of various knowledge. Further analysis shows that our approach has the ability to derive the meaning representation of a context-dependent utterance by leveraging previously generated outcomes.

Keywords: semantic parsing · question answering.

1 Introduction

We consider the problem of table-based conversational question answering, which is crucial for allowing users to interact with web tables or a relational databases using natural language. Given a table, a question/utterance[4] and the history of an interaction, the task calls for understanding the meanings of both current and historical utterances to produce the answer. In this work, we tackle the problem in a semantic parsing

[4] In this work, we use the terms "*utterance*" and "*question*" interchangeably.



Year	City	Country	Nations
1896	Athens	Greece	14
1900	Pairs	France	24
2004	Athens	Greece	201
2008	Beijing	China	204
2012	London	UK	204

1st turn
Q1: Which city hosted the Summer Olympics in 2008?
SQL1: SELECT City WHERE Year = 2008
A1: Beijing

2nd turn
Q2: How many nations participated that year?
SQL2: SELECT Nations WHERE Year = 2008
A2: 204

3rd turn
Q3: How about 2004?
SQL3: SELECT Nations WHERE Year = 2004
A3: 201

Fig. 1. A running example that illustrates the input and the output of the problem.

paradigm [23, 27, 30, 12]. User utterances are mapped to their formal meaning representations/logical forms (e.g. SQL queries), which could be regarded as programs that are executed on a table to yield the answer. We use SequentialQA [8] as a testbed, and follow their experiment settings which learn from denotations (answers) without access to the logical forms.

The task is challenging because successfully answering a question requires understanding the meanings of multiple inputs and reasoning based on that. A model needs to understand the meaning of a question based on the meaning of a table as well as the understanding about historical questions. Take the second turn question ("*How many nations participate in that year?*") in Figure 1 as an example. The model needs to understand that the question is asking about the number of nations with a constraint on a particular year. Here the year ("*2008*") is not explicitly in Q2, but is a carry over from the analyzed result of the previous utterance. There are different types of ellipsis and co-reference phenomena in user interactions. The missing information in Q2 corresponds to the previous WHERE condition, while the missing part in Q3 comes from the previous SELECT clause. Meanwhile, the whole process is also on the basis of understanding the meaning of a table including column names, cells, and the relationships between column names and cells.

Based on the aforementioned considerations, we present a conversational table-based semantic parser, abbreviated as CAMP, by introducing various types of knowledge in this work, including grammar knowledge, expert knowledge, and external resource knowledge. First, we introduce grammar knowledge, which is the backbone of our model. Grammar knowledge includes a set of actions which could be easily used for reasoning and leveraging historical information. We extend the grammar of [8], so that the model has the ability to copy logical form segment from previous outputs. Therefore, our model effectively handles the co-reference and ellipsis phenomena in conversation, as shown in the second and third turns in Figure 1. The grammar knowledge also help us design strategy in training data collection phase to prune spurious logical forms. Second, we use the expert knowledge to help us design model structure. Considering that a decomposable model is more controllable, we decompose the entire pipeline into submodules which are coupled with the predefined actions in the grammar closely. This further enables us to sample valid logical forms with improved heuristics, and learn submodules with fine-grained supervision. Third, we introduce several kinds of external resource knowledge to improve the understanding of input semantic meanings. For a better question representation, we take advantage of a pre-trained language model by leveraging a large unstructured text corpus. For a better table representation, we use several lexical analysis datasets and use the pre-trained models to give each table header semantic type information, i.e., NER type.

We train model parameters from denotations without access to labeled logical forms, and conduct experiments on the SequentialQA dataset [8]. Results show that our model achieves state-of-the-art accuracy. Further analysis shows that (1) incrementally incorporating various types of knowledge could bring performance boost, and (2) the model is capable of replicating previously generated logical forms to interpret the logical form of a conversational utterance.

2 Grammar

Action	Operation	# Arguments
A1	SELECT-Col	# columns
A2	WHERE-Col	# columns
A3	WHERE-Op	# operations
A4	WHERE-Val	# valid cells
A5	COPY SELECT	1
A6	COPY WHERE	1
A7	COPY SELECT + WHERE	1

Table 1. Actions and the number of action instances in each type. Operations consist of $=, \neq, >, \geq, <, \leq, argmin, argmax$. A1 means selecting an column in SELECT expression. A2, A3 and A4 means selecting a column, a operation and a cell value in WHERE expression. A3 means select a condition operation, A4 means select a cell value, A5 means copying the previous SELECT expression, A6 means copying the previous WHERE expression, A7 means copying the entire previous SQL expression.

Partly inspired by the success of the sequence-to-action paradigm [7, 2, 25] in semantic parsing, we treat the generation of a logical form as the generation of a linearized action sequence following a predefined grammar. We use a SQL-like language as the logical form, which is a standard executable language on web tables. Each query of this logical form language consist of one SELECT expression and zero or one WHERE expression. The SELECT expression shows which column can be chosen and the WHERE expression add a constraint on which row the answer can be chosen. The SELECT expression consists of key word **SELECT** and a column name. The WHERE expression, which starts with the key word **WHERE**, consists of one or more condition expressions joined by the key word **AND**. Each condition expression consists of a column name, an operator, and a value. Following [8], we consider the following operators: $=, \neq, >, \geq, <, \leq, argmin, argmax$. Since most of the questions in the SequtialQA dataset are simple questions, we do not consider the key word AND in this work, which means the WHERE expression only contains one condition expression.

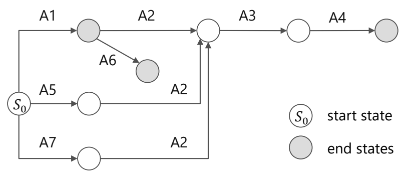


Fig. 2. Possible action transitions based on our grammar as described in Table 1.

We describe the actions of our grammar in Table 1. The first four actions (A1-A4) are designed to infer the logical forms based on the content of current utterance. The last three actions are designed to replicate the previously generated logical forms. The number of arguments in copying actions are all equals to one because the SequtialQA dataset is composed of simple questions. Our approach can be easily extend to complex questions by representation previous logical form segment with embedding vector [25]

3 Approach

Given a question, a table, and previous questions in a conversation as the input, the model outputs a sequence of actions, which is equivalent to a logical form (i.e. SQL query in this work) which is executed on a table to obtain the answer. Figure 3 show the overview of our approach. After encoding the questions into the vector representation, we first use a controller module to predict a sketch (which we will describe later) of the action sequence. Afterwards we use modules to predict the argument of each action in the sketch. We will describe the controller and these modules in this section, and will also show how to incorporate knowledge these modules.

Question Encoder We describe how we encode the input question to a vector representation in this part.

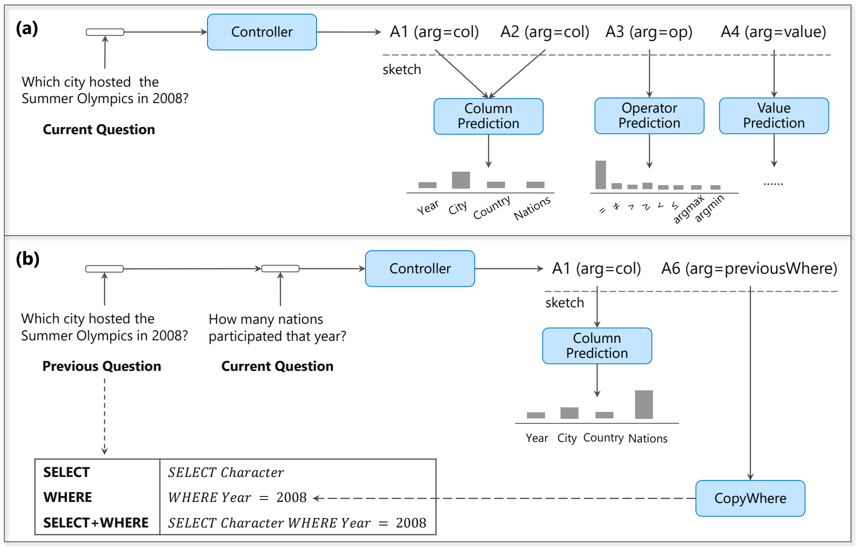
The input includes the current utterance and the utterance in the previous turn. We concatenate them with a special splitter and map each word to a continuous embedding vector $\mathbf{x}_t^I = \mathbf{W}_x \mathbf{o}(x_t)$, where $\mathbf{W}_x \in \mathbb{R}^{n \times |\mathcal{V}_x|}$ is an embedding matrix, $|\mathcal{V}_x|$ is the vocabulary size, and $\mathbf{o}(x_t)$ a one-hot vector.

We use a bi-directional recurrent neural network with gated recurrent units (GRU) [4] to represent the contextual information of each word. The encoder computes the hidden vectors in a recursive way. The calulation of the t-th time step is given as follows:

$$\overrightarrow{\mathbf{e}}_t = \mathrm{f}_{\mathrm{GRU}}\left(\overrightarrow{\mathbf{e}}_{t-1}, \mathbf{x}_t\right), t = 1, \cdots, |x| \tag{1}$$

$$\overleftarrow{\mathbf{e}}_t = \mathrm{f}_{\mathrm{GRU}}\left(\overleftarrow{\mathbf{e}}_{t+1}, \mathbf{x}_t\right), t = |x|, \cdots, 1 \tag{2}$$

$$\mathbf{e}_t = [\overrightarrow{\mathbf{e}}_t, \overleftarrow{\mathbf{e}}_t] \tag{3}$$



SELECT	*SELECT Character*
WHERE	*WHERE Year = 2008*
SELECT+WHERE	*SELECT Character WHERE Year = 2008*

Fig. 3. Overview of our model architecture. Part (a) shows how we handle a single turn question. Part (b) shows how we handle question that replicates logical form segment from the previous utterance.

where $[\cdot,\cdot]$ denotes vector concatenation, $\mathbf{e}_t \in \mathbb{R}^n$, and $\mathrm{f}_{\mathrm{GRU}}$ is the GRU function. We denote the input x as: $\tilde{\mathbf{e}} = [\overrightarrow{\tilde{\mathbf{e}}}_{|x|}, \overleftarrow{\tilde{\mathbf{e}}}_1]$

Inspired by the recently success of incorporating contextual knowledge in a variety of NLP tasks [22, 21], we further enhance the contextual representation of each word by using a language model which is pretrained on a external text corpus. In this work, we train a bidirectional language model from Paralex [6], which includes 18 million question-paraphrase pairs scraped from WikiAnswers. The reason why we choose this dataset is that the question-style texts are more consistent with the genre of our input. For each word x, we concat the word representaion and the hidden state LM_t of the pretrained language model. $\mathbf{x}_t = [\mathbf{x}_t^I, LM_t]$

Table Encoder In this part, we describe how we encode headers and table cells into vector representations.

A table consists of M headers (column names)[5] and $M * N$ cells where N is the number of rows. Each column consists of a header and several cells. Let us denote the k-th headers as c_k. Since a header may consist of multipule words, we use GRU RNN to calculate the presentation of each words and use use the last hidden state as the header vector representation $\{\mathbf{c}_k\}_{k=1}^{M}$. Cell values could be calculated in the same way.

We further improve header representation by considering typing information of cell-s. The reason is that incorporating typing information would improve the predication of a header in SELECT and WHERE expressions. Take Q2 in Figure 1 as an example.

[5] In this work, we use the terms "*header*" and "*column name*" interchangeably.

People can infer that the header "Nation" is talking about number because the cells in the same column are all number, and can use this information to better match to the question starting with "how many". Therefore, the representation of a header not only depends on the words it contains, but also relates to the cells under the same column. Specifically, we use Stanford CoreNLP [14] to get the NER result of each cell, and then use an off-the-shell mapping rule to get the type of each cell [10]. The header type is obtained by voting from the types of cells. We follow [10] and set the types as {COUNTRY, LOCATION, PERSON, DATE, YEAR, TIME, TEXT, NUMBER, BOOLEAN, SEQUENCE, UNIT}.

Formally, every header $\mathbf{c}_k$ also has a continuous type representation via $\mathbf{t}_k = \mathbf{W}_t \mathbf{o}\left(c_t\right)$, where $\mathbf{W}_t \in \mathbb{R}^{n \times |\mathcal{V}_t|}$ is an embedding matrix, $|\mathcal{V}_t|$ is the total number of type, and $\mathbf{o}\left(c_t\right)$ a one-hot vector. The final representation of a header is the concatenation of word-based vector and type-based vector $\mathbf{t_k}$, which we denote as follows.

$$\tilde{\mathbf{c_k}} = [\mathbf{c}_k, \mathbf{t}_k] \tag{4}$$

Controller Given a current and previous question as input, the controller predict a sketch which is an action sequence without arguments between a starting state and an ending state. As the number of all possible sketches we define is small, we model sketch generation as a classification problem. Specifically, the sketch of $[A1]$ means a logical form that only contains a SELECT expression, which is inferred based on the content of the current utterance. The sketch of $[A1 \rightarrow A2 \rightarrow A3 \rightarrow A4]$ means a logical form having both SELECT expression and WHERE expression, in which case all the arguments are inferred based on the content of the current utterance. The sketch of $[A5 \rightarrow A2 \rightarrow A3 \rightarrow A4]$ stands for a logical form that replicates the SELECT expression of the previous utterance and infer out other constituents based on the current utterance. The sketch of $[A6 \rightarrow A2 \rightarrow A3 \rightarrow A4]$ represents a logical form that replicates previous the WHERE expression, and get the SELECT expression based on the current utterance. The sketch of $[A7 \rightarrow A2 \rightarrow A3 \rightarrow A4]$ means a logical form that replicates both SELECT expression and WHERE expression of the previous utterance, and incorporate additional constraint as another WHERE expression based on the current utterance. Similar strategy has been proven effective in single-turn sequence-to-SQL generation [5].

Formally, given the current question x_{cur} and the previous question x_{pre}, we use Equations 3 to get their vector representation $\tilde{\mathbf{e}}_{cur}$ and $\tilde{\mathbf{e}}_{pre}$. We treat each sketch s as a category, and use a softmax classifier to compute $p\left(s|x\right)$ as follows, where $\mathbf{W}_a \in \mathbb{R}^{|\mathcal{V}_s| \times 2n}$, $\mathbf{b}_a \in \mathbb{R}^{|\mathcal{V}_s|}$ are parameters.

$$p\left(s|x\right) = \mathrm{softmax}_s\left(\mathbf{W}_s[\tilde{\mathbf{e}}_{cur}, \tilde{\mathbf{e}}_{pre}] + \mathbf{b}_s\right)$$

Column Prediction For action A1 and A2, we build two same neural models with different parameters. Both of them are used for predicting a column, just one in SELECT expressions and one in WHERE expressions. We encode the input question x into $\{\mathbf{e}_t\}_{t=1}^{|x|}$ using GRU units. For each column, we employ the column attention mechnism [28] to capture most relevant information from question. The column-aware question information is useful for column prediction. Specifically, we use an attention mechanism towards question vectors $\{\mathbf{e}_t\}_{t=1}^{|x|}$ to obtain the column-specific representation for $\mathbf{c}_k$.

The attention score from $\mathbf{c}_k$ to $\mathbf{e}_t$ is computed via $u_{k,t} \propto \exp\{\alpha(\mathbf{c}_k) \cdot \alpha(\mathbf{e}_t)\}$, where $\alpha(\cdot)$ is a one-layer neural network, and $\sum_{t=1}^{M} u_{k,t} = 1$. Then we compute the context vector $\mathbf{e}_k^c = \sum_{t=1}^{M} u_{k,t}\mathbf{e}_t$ to summarize the relevant question words for $\mathbf{c}_k$.

We calculate the probability of each column $\mathbf{c}_k$ via

$$\sigma(\mathbf{x}) = \mathbf{w}_3 \cdot \tanh\left(\mathbf{W}_4\mathbf{x} + \mathbf{b}_4\right) \tag{5}$$

$$p\left(\texttt{col} = k|x\right) \propto \exp\{\sigma([[\tilde{\mathbf{e}}, \mathbf{e}_k^c], \mathbf{c}_k])\} \tag{6}$$

where $\sum_{j=1}^{M} p\left(\texttt{col} = j|x\right) = 1$, and $\mathbf{W}_4 \in \mathbb{R}^{3n \times m}$, $\mathbf{w}_3, \mathbf{b}_4 \in \mathbb{R}^m$ are parameters.

Operator Prediction In this part, we need to predict an operator from the list $[=, \neq, >, \geq, <, \leq, argmin, argmax]$. We regard this task as a classification problem and use the same neural architecture in the controller module to make prediction. For implementation, we randomly initialize the parameters and set the softmax's prediction category to the number of our operators, which is equals to 8 in this work.

Value Prediction We prediction WHERE value based on two evidences. The first one comes from a neural network model which has the same architecture as the one used for column prediction. The second ones is calculated based on the number of word overlap between cell words and question words. We incorporate the second score because we observe that many WHERE values are table cells that have string overlap with the question. For example in Figure 1, both the first and the third questions fall into this category. Formerly, the final probability of a cell to be predicted is calculated as a linear combination of both distributions as following,

$$p\left(\texttt{cell} = k|x\right) = \lambda\hat{p}\left(\texttt{cell} = k|x\right) + (1 - \lambda)\alpha_k^{cell} \tag{7}$$

where $\hat{p}\left(\texttt{cell} = k|x\right)$ is the probability distribution obtained from the neural network and α_k^{cell} is the overlapping score normalized by softmax and λ is a hyper parameter.

COPYING Action Prediction As described in table 1, we have tree copy-related actions (i.e. A5, A6, A7) to predict which component in the previous logical form should be copied to the current logical form. In this work this functionality is achieved by the controller model because the SequentialQA dataset only contains simple questions whose logical forms do not contain more than one WHERE expressions. Our model easily extend to copy logical form segments from complex questions. An intuitive way to achieve this goal is representing each logical form component as a vector representation and applying an attention mechanism to choose the most relevant logical form segment [25].

4 Training Data Collection

The SequentialQA dataset only provides question-denotation pairs, while our model requires question-action sequence pairs as the training data. Therefore, we use the following strategies to automatically generate the logical form for each question, which is equivalent to an action sequence. For acquiring the logical forms which are not provided by the SequentialQA dataset, we traverse the valid logical form space using breadth-first search following the action transition graph as illustrated in Figure 2. For the purpose of preventing combinatorial explosion in searching pace, we prune the search space by executing the partial semantic parse over the table to get answers during the search process. In this way, a path could be filtered out if its answers have no overlap with the golden answers.

We use two strategies to handle the problem of spurious logical forms and to favor the actions of replicating from previous logical form segments, respectively. The first strategy (**S1**) is for pruning spurious logical forms. Spurious logical forms could be executed to get the correct answer but do not reflect the semantic meaning in the question. Pruning logical forms is vital for improving model performance according to previous studies [18, 15]. We only keep those logic forms whose components in the where clause have word overlap with the words in questions. The second strategy (**S2**) is for encouraging the model to learn sequential aspects of the dataset. We only keep the logical form with the COPY action after pruning spurious logical forms.

5 Experiment

We conduct the experiments on the SequentialQA dataset which has 6,066 unique questions sequences containing 17,553 total question-answer pairs (2.9 questions per sequence). The dataset is divided into train and test in an 83%/17% split. We optimize our model parameters using standard stochastic gradient descent. We represent each word using word embedding [20] and the mean of the sub-word embeddings of all the n-grams in the word (Hashimoto et al., 2016). The dimension of the concatenated word embedding is 400. We clamp the embedding values to avoid over-fitting. We set the dimension of hidden state as 200, the dimension of type representation as 5, set the batch size as 32, and the dropout rate as 0.3. We initialize model parameters from a uniform distribution with fan-in and fan-out. We use Adam as our optimization method and set the learning as 0.001.

5.1 Baseline Systems

We describe our baseline systems as follows. **Floating Parser** [17] builds a system which first generates logical forms using a floating parser (FP) and then ranks the generated logical forms with a feature-based model. FP is like traditional chart parser but designed with specific deduction rules to alleviate the dependency on a full-fledged lexicon. **Neural Programmer**(NP) [16] is an end-to-end neural network-based approach. NP has a set of predefined operations, and Instead of directly generating a logical form, this system outputs a program consists of a fixed length of operations on a table. The handcraft operations is selected via attention mechanism [3] and the history information is conveyed by an RNN. **DynSP** [8] constructs the logical form by applying predefined actions. It learns the model from annotations in a trial-and-error paradigm. The model is trained with policy functions using an improved algorithm proposed by [19]. DynSP* stands for an improved version that better utilize contextual information.

Model	ALL	SEQ	POS 1	POS 2	POS 3
FP	34.1	7.2	52.6	25.6	**25.9**
NP	39.4	10.8	58.9	35.9	24.6
DynSP	42.0	10.2	70.9	35.8	20.1
FP+	33.2	7.7	51.4	22.2	22.3
NP+	40.2	11.8	60.0	35.9	25.5
DynSP*	44.7	12.8	70.4	41.1	23.6
CAMP	45.0	11.7	71.3	42.8	21.9
CAMP + TU	**45.5**	12.7	**71.1**	**43.2**	22.5
CAMP + TU + LM	**45.5**	**13.2**	70.3	42.6	24.8

Table 2. Accuracies of all systems on SequentailQA; the models in the top section of the table treat questions independently, while those in the middle consider sequential context. Our method in the bottom section also consider sequential context and outperforms existing ones both in terms of overall accuracy as well as sequence accuracy

The experiment results related to FP and NP are reported by [8]. The details of how they adjust these two models to the SequentialQA dataset can be seen in their paper. We implement there variants of our model for comparision: CAMP is our basic framework where no external knowledge are used. In CAMP + TU, we incorporate type knowledge in table understanding module. In CAMP + TU + LM, we further use contextual knowledge in question representation module.

5.2 Results and Analysis

MODELS	CAMP	+ TU	+ TU + LM
CONTROLLER	83.5	83.5	84.8
SELECT-COL	82.5	83.4	83.7
WHERE-COL	35.0	35.9	36.6
OPERATION	69.7	69.7	70.2
VALUE	21.2	21.2	21.5

Table 3. Accuracy for each module in different settings.

We can see the result of all baseline systems as well as our method in Table 2. We show accuracy for all questions, for each sequence (percentage of the correctly answered sequences of sentence), and for each sentence in particular position of a sequence. We can see that CAMP performers better than existing systems in terms of overall accuracy. Adding table knowledge improves overall accuracy and sequence accuracy. Adding contextual knowledge further improve the sequence accuracy.

We study the performance of each module of CAMP. From Table 3 we can see that table knowledge and contextual knowledge both bring improvements in these modules. The controller module and the column prediction module in SELECT expression achieves higher accuracies. The reason is that, compared to other modules, the supervise signals for these two modules are less influenced by spurious logical forms. Compared to the WHERE part, the SELECT part of a SQL query is less spurious because the answer typically has the same column as the SELECT part, while the WHERE part are more uncontrollable.

We conduct error analysis to understand the limitation of our approach and shed light on future directions. The errors are mainly caused by error propagation and semantic matching problems and limitation of our grammar. . After counting the numbers in 100 false predictions for the test set, we estimate that there are 15% of them for error propagation; 28% of them for semantic matching problems; 21% of them for limitation of our grammar; 36% of them for other reasons.

6 Related Work

This work closely relates to two lines of work, namely table-based semantic parsing and context-dependent semantic parsing. We describe the connections and the differences in this section.

Table-based semantic parsing aims to map an utterance to an executable logical form, which can be considered a program to execute on a table to yield the answer [17, 9, 11]. The majority of existing studies focus on single-turn semantic parsing, in which case the meaning of the input utterance is independent of the historical interactions. Existing studies on single-turn semantic parsing can be categorized based on the type of supervision used for model training. The first category is the supervised setting, in which case the target logical forms are explicitly provided. Supervised learning models including various sequence-to-sequence model architectures and slot filling based models have proven effective in learning the patterns involved in this type of parallel training data. The second category is weak supervised learning, in which case the model can only access answers/denotations but does not have the annotated logical forms. In this scenario, logical forms are typically regarded as the hidden variables/states. Maximum marginal likelihood and reinforcement learning have proven effective in training the model [7]. Semi supervised learning is also investigated to further consider external unlabeled text corpora [29]. Different from the aforementioned studies, our work belongs to multi-turn table-based semantic parsing. The meaning of a question also depends on the conversation history. The most relevant work is [8], the authors of which also develop the SequentialQA dataset. We have described the differences between our work and the work of [8] in the introduction section.

In context-dependent semantic parsing, the understanding of an utterance also depends on some contexts. We divide existing works based on the different types of "context", including the historical utterances and the state of the world which is the environment to execute the logical form on. Our work belongs to the first group, namely historical questions as the context. In this field, [30] learn the semantic parser from annotated lambda-calculus for ATIS flight planning interactions. They first carry out context-independent parsing with Combinatory Categorial Grammar (CCG), and then resolve all references and optionally perform an elaboration or deletion. [26] deal with an interactive tourist information system, and use a set of classification modules to predict different arguments. [25] also study on ATIS flight planning datasets, and introduce an improved sequence-to-sequence learning model to selectively replicate previous logical form segments. In the second group, the world is regarded as the context and the state of the world is changeable as actions/logical forms are executed on the world. [1] focus on spatial instructions in the navigation environment and train a weighted C-CG semantic parser. [13] build three datasets including ALCHEMY, TANGRAMS and SCENE domains. They take the starting state and the goal state of the entire instructions, and develop a shift-reduce parser based on a defined grammar for each domain. [24] further introduce a learning algorithm to maximize the immediate expected rewards for all possible actions of each visited state.

7 Conclusion

In this work, we present a conversational table-based semantic parser called CAMP that integrates various types of knowledge. Our approach integrates various types of knowledge, including unlabeled question utterances, typing information from external resource, and an improved grammar which is capable of replicating previously predicted action subsequence. Each module in the entire pipeline can be conventionally improved. We conduct experiments on the SequentialQA dataset, and train the model from question-denotation pairs. Results show that incorporating knowledge improves the accuracy of our model, which achieves state-of-the-art accuracy on this dataset. Further analysis shows that our approach has the ability to discovery and utilize previously generated logical forms to understand the meaning of the current utterance.

References

1. Artzi, Y., Zettlemoyer, L.: Weakly supervised learning of semantic parsers for mapping instructions to actions. Transactions of the Association of Computational Linguistics **1**, 49–62 (2013)
2. Chen, B., Han, X., Su, L.: Sequence-to-action: End-to-end semantic graph generation for semantic parsing. In: ACL (2018)
3. Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder–decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1724–1734. Association for Computational Linguistics (2014). https://doi.org/10.3115/v1/D14-1179, `http://www.aclweb.org/anthology/D14-1179`
4. Cho, K., van Merrienboer, B., ?aglar Gül?ehre, Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. In: EMNLP (2014)
5. Dong, L., Lapata, M.: Coarse-to-fine decoding for neural semantic parsing. In: ACL (2018)
6. Fader, A., Zettlemoyer, L., Etzioni, O.: Paraphrase-driven learning for open question answering. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). vol. 1, pp. 1608–1618 (2013)

7. Guu, K., Pasupat, P., Liu, E., Liang, P.: From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. pp. 1051–1062 (2017)
8. Iyyer, M., Yih, W.t., Chang, M.W.: 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). pp. 1821–1831. Association for Computational Linguistics (2017). https://doi.org/10.18653/v1/P17-1167, `http://www.aclweb.org/anthology/P17-1167`
9. Krishnamurthy, J., Dasigi, P., Gardner, M.: Neural semantic parsing with type constraints for semi-structured tables. In: EMNLP (2017)
10. Li, X., Roth, D.: Learning question classifiers. In: COLING (2002)
11. Liang, C., Norouzi, M., Berant, J., Le, Q., Lao, N.: Memory augmented policy optimization for program synthesis with generalization. arXiv preprint arXiv:1807.02322 (2018)
12. Liang, P.: Learning executable semantic parsers for natural language understanding. Communications of the ACM **59**(9), 68–76 (2016)
13. Long, R., Pasupat, P., Liang, P.: Simpler context-dependent logical forms via model projections. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics. pp. 1456–1465 (2016)
14. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J.R., Bethard, S., McClosky, D.: The stanford corenlp natural language processing toolkit. In: ACL (2014)
15. Mudrakarta, P.K., Taly, A., Sundararajan, M., Dhamdhere, K.: It was the training data pruning too! CoRR **abs/1803.04579** (2018)
16. Neelakantan, A., Le, Q.V., Abadi, M., McCallum, A., Amodei, D.: Learning a natural language interface with neural programmer. In: Proceedings of the International Conference on Learning Representations (2017)
17. Pasupat, P., Liang, P.: Compositional semantic parsing on semi-structured tables. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics. pp. 1470–1480 (2015)
18. Pasupat, P., Liang, P.: Inferring logical forms from denotations. CoRR **abs/1606.06900** (2016)
19. Peng, H., Chang, M.W., tau Yih, W.: Maximum margin reward networks for learning from explicit and implicit supervision. In: EMNLP (2017)
20. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543 (2014)
21. Peters, M.E., Neumann, M., Zettlemoyer, L., Yih, W.t.: Dissecting Contextual Word Embeddings: Architecture and Representation. ArXiv e-prints (Aug 2018)
22. Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics. pp. 2227–2237 (2018)
23. Prolog, P.: Learning to parse database queries using inductive logic programming (1996)
24. Suhr, A., Artzi, Y.: Situated mapping of sequential instructions to actions with single-step reward observation. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics. pp. 2072–2082. Association for Computational Linguistics (2018)
25. Suhr, A., Iyer, S., Artzi, Y.: Learning to map context-dependent sentences to executable formal queries. arXiv preprint arXiv:1804.06868 (2018)
26. Vlachos, A., Clark, S.: A new corpus and imitation learning framework for context-dependent semantic parsing. Transactions of the Association for Computational Linguistics **2**, 547–559 (2014)
27. Wong, Y.W., Mooney, R.J.: Learning synchronous grammars for semantic parsing with lambda calculus. In: Annual Meeting-Association for computational Linguistics. p. 960. No. 1 (2007)

28. Xu, X., Liu, C., Song, D.X.: Sqlnet: Generating structured queries from natural language without reinforcement learning. CoRR **abs/1711.04436** (2017)
29. Yin, P., Zhou, C., He, J., Neubig, G.: Structvae: Tree-structured latent variable models for semi-supervised semantic parsing. arXiv preprint arXiv:1806.07832 (2018)
30. Zettlemoyer, L.S., Collins, M.: 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. pp. 976–984 (2009)