Memory-Based Semantic Parsing

Parag Jain and Mirella Lapata
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB, UK
parag.jain@ed.ac.uk mlap@inf.ed.ac.uk

Abstract

We present a memory-based model for context-dependent semantic parsing. Previous approaches focus on enabling the decoder to copy or modify the parse from the previous utterance, assuming there is a dependency between the current and previous parses. In this work, we propose to represent contextual information using an external memory. We learn a context memory controller that manages the memory by maintaining the cumulative meaning of sequential user utterances. We evaluate our approach on three semantic parsing benchmarks. Experimental results show that our model can better process context-dependent information and demonstrates improved performance without using task-specific decoders.

1 Introduction

Semantic parsing is the task of converting natural language utterances into machine interpretable meaning representations such as executable queries or logical forms. It has emerged as an important component in many natural language interfaces (Őzcan et al., 2020) with applications in robotics (Dukes, 2014), question answering (Zhong et al., 2018; Yu et al., 2018b), dialogue systems (Artzi and Zettlemoyer, 2011), and the Internet of Things (Campagna et al., 2017).

Neural network based approaches have led to significant improvements in semantic parsing (Zhong et al., 2018; Kamath and Das, 2019; Yu et al., 2018b; Yavuz et al., 2018; Yu et al., 2018a) across domains and semantic formalisms. The majority of existing studies focus on parsing utterances in isolation, and as a result they cannot readily transfer in more realistic settings where users ask multiple inter-related questions to satisfy an information need. In this work, we study *context-dependent* semantic parsing focusing specifically on text-to-SQL generation, which has emerged as a popular application area in recent years.

Figure 1 shows a sequence of utterances in an interaction. The discourse focuses on a specific *topic* serving a specific information need, namely, finding out which Continental flights leave from Chicago on a given date and time. Importantly, interpreting each of these utterances, and mapping them to a database query to retrieve an answer, needs to be situated in a particular context as the exchange proceeds. The topic further evolves as the discourse transitions from one utterance to the next and constraints (e.g., TIME or PLACE) are added or revised. For example, in Q2 the TIME constraint *before 10am* from Q1 is revised to *before noon*, and in Q3 to *before 2pm*. Aside from such *topic extensions* (Chai and Jin, 2004), the interpretation of Q2 and Q3 depends on Q1, as it is implied that the questions concern Continental flights that go from Chicago to Seattle, not just any Continental flights, however the phrase *from Chicago to Seattle* is elided from Q2 and Q3. The interpretation of Q4 depends on Q3, which in turn depends on Q1. Interestingly, Q5 introduces information with no dependencies on previous discourse and, in this case, relying on information from previous utterances will lead to incorrect SQL queries.

The problem of contextual language processing has been most widely studied within dialogue systems where the primary goal is to incrementally fill pre-defined slot-templates, which can be then used to generate appropriate natural language responses (Gao et al., 2019). But the rich semantics of SQL queries makes the task of contextual text-to-SQL parsing substantially different. Previous approaches (Suhr et al., 2018; Zhang et al., 2019) tackle this problem by enabling the decoder to copy or modify the *previous* queries under the assumption that they contain all necessary context for generating the current SQL query. The

Transactions of the Association for Computational Linguistics, vol. 9, pp. 1197–1212, 2021. https://doi.org/10.1162/tacl_a_00422
Action Editor: Mike Lewis. Submission batch: 4/2021; Revision batch: 6/2021; Published 11/2021.

Q1: What Continental flights go from Chicago to Seattle before 10 am in morning 1993 February twenty sixth

SQL1: (SELECT DISTINCT flight.flight_id FROM flight WHERE (flight.airline_code = ’CO’ AND (flight . from_airport IN (SELECT airport_service . airport_code FROM airport_service WHERE airport_service . city_code IN (SELECT city . city_code FROM city WHERE city.city_name = ’CHICAGO’)) AND (flight . to_airport IN (SELECT airport_service . airport_code FROM airport_service WHERE airport_service . city_code IN (SELECT city . city_code FROM city WHERE city.city_name = ’SEATTLE’)) AND (flight.departure_time < 1000))))) ;

Q2: Continental flights before noon that have a meal

Q3: Continental flights before 2 pm

Q4: On 1993 February twenty seventh

Q5: All Continental flights leaving Chicago before 8 am on 1993 February twenty seventh

Figure 1: Example utterances from a user interaction in the ATIS dataset. Utterance segments referring to the same entity or objects are in same color. SQL queries corresponding to Q2–Q5 follow a pattern similar to Q1 and are not shown for the sake of brevity.

utterance history is encoded in a hierarchical manner and although this is a good enough approximation for most queries (in existing datasets), it is not sufficient to model long-range discourse phenomena (Grosz and Sidner, 1986).

Our own work draws inspiration from Kintsch and van Dijk’s (1978) text comprehension model. In their system the process of comprehension involves three levels of operations. Firstly, smaller units of meaning (i.e., propositions) are extracted and organized into a coherent whole (*microstructure*); some of these are stored in a working memory buffer and allow to decide whether new input overlaps with already processed propositions. Secondly, the gist of the whole is condensed (*macrostructure*). And thirdly, the previous two operations generate new texts in working with the memory. In other words, the (short and long term) memory of the reader gives meaning to the text read. They propose three macro rules, namely, deletion, generalization, and construction, as essential to reduce and organize the detailed information of the microstructure of the text. Furthermore, previous knowledge and experience are central to the interpretation of text enabling the reader to fill information gaps.

Our work borrows several key insights from Kintsch and van Dijk (1978) without being a direct implementation of their model. Specifically, we also break down input utterances into smaller units, namely, phrases, and argue that this information can be effectively utilized in maintaining contextual information in an interaction. Furthermore, the notion of a *memory* buffer that can be used to store and process new and old information plays a prominent role in our approach. We propose a **Mem**ory-based **C**ont**E**xt model (which we call MemCE for short) for keeping track of contextual information, and learn a context memory controller that manages the memory. Each interaction (sequence of user utterances) maintains its context using a memory matrix. User utterances are segmented into a sequence of phrases representing either new information to be added into the memory (e.g., *that have a meal* in Figure 1) or old information which might conflict with current information in memory and needs to be updated (e.g., *before 10 am* should be replaced with *before noon* in Figure 1). Our model can inherently add new content to memory, read existing content by accessing the memory, and update old information.

We evaluate our approach on the ATIS (Suhr et al., 2018; Dahl et al., 1994), SParC (Yu et al., 2019b), and CoSQL (Yu et al., 2019a) datasets. We observe performance improvements when we combine MemCE with existing models underlying the importance of more specialized mechanisms for processing context information. In addition, our model brings interpretability in how the context is processed. We are able to inspect the learned memory controller and analyze whether important discourse phenomena such as coreference and ellipsis are modeled.

2 Related Work

Sequence-to-sequence neural networks (Bahdanau et al., 2015) have emerged as a general modeling framework for semantic parsing, achieving impressive results across different domains and semantic formalisms (Dong and Lapata, 2016; Jia and Liang, 2016; Iyer et al., 2017; Wang et al., 2020; Zhong et al., 2018; Yu et al., 2018b, inter alia). The majority of existing work has focused on mapping natural language utterances into machine-readable meaning representations *in isolation* without utilizing context information. While this is useful for environments consisting of one-shot interactions of users with a system (e.g., running QA queries on a database), many settings require extended interactions between a user and an automated assistant (e.g., booking a flight). This makes the one-shot parsing model inadequate for many scenarios.

In this paper we are concerned with the lesser studied problem of *contextualized* semantic parsing where previous utterances are taken into account in the interpretation of the current utterance. Earlier work (Miller et al., 1996; Zettlemoyer and Collins, 2009; Srivastava et al., 2017) has focused on symbolic features for representing context—for example, by explicitly modeling discourse referents, or the flow of discourse. More recent neural methods extend the sequence-to-sequence architecture to incorporate contextual information either by modifying the encoder or the decoder. Context-aware encoders resort to concatenating the current utterance with the utterances preceding it (Suhr et al., 2018; Zhang et al., 2019) or focus on the history of the utterances most relevant to the current decoder state (Liu et al., 2020). The decoders take context representations as additional input and often copy segments from the previous query (Suhr et al., 2018; Zhang et al., 2019). Hybrid approaches (Iyyer et al., 2017; Guo et al., 2019; Liu et al., 2020; Lin et al., 2019) employ neural networks for representation learning but use a grammar for decoding (e.g., a sequence of actions or an intermediate representation).

A tremendous amount of work has taken place in the context of discourse modeling focusing on extended texts (Mann and Thompson, 1988; Hobbs, 1985) and dialogue (Grosz and Sidner, 1986). Kintsch and van Dijk (1978) study the mental operations underlying the comprehension and summarization of text. They introduce *propositions* as the basic unit of text representation, and a model of how incoming text is processed given memory limitations; texts are reduced to important propositions (to be recalled later) using *macro-operators* (e.g., addition, deletion). Their model has met with popularity in cognitive psychology (Baddeley, 2007) and has also found application in summarization (Fang and Teufel, 2016).

Our work proposes a new encoder for contextualized semantic parsing. At the heart of our approach is a memory controller that keeps track of context via writing new information and updating old information. Our memory-based approach is inspired by Kintsch and van Dijk (1978) and is closest to Santoro et al. (2016), who use a memory augmented neural network (Weston et al., 2015; Sukhbaatar et al., 2015) for meta-learning. Specifically, they introduce a method for accessing external memory which functions as short-term storage for meta-learning. Although we report experiments solely on semantic parsing, our encoder is fairly general and could be applied to other context-dependent tasks such as conversational information seeking (Dalton et al., 2020) and information retrieval (Sun and Chai, 2007; Voorhees, 2004).

3 Model

Our model is based on the encoder-decoder architecture (Cho et al., 2015) with the addition of a memory component (Sukhbaatar et al., 2015; Santoro et al., 2016) for incorporating context. Let $I = [X_i, Y_i]_{i=1}^{n}$ denote an interaction such that X_i is the input utterance and Y_i is the output SQL at interaction turn $I[i]$. At each turn i, given X_i and all previous turns $I[1 \ldots i-1]$, our task is to predict SQL output Y_i.

As shown in Figure 2, our model consists of four components: (1) a memory matrix retains discourse information, (2) a memory controller, which learns to access and manipulate the memory such that correct discourse information is retained, (3) utterance and phrase encoders, and (4) a decoder that interacts with the memory and utterance encoder using an attention mechanism to generate SQL output.

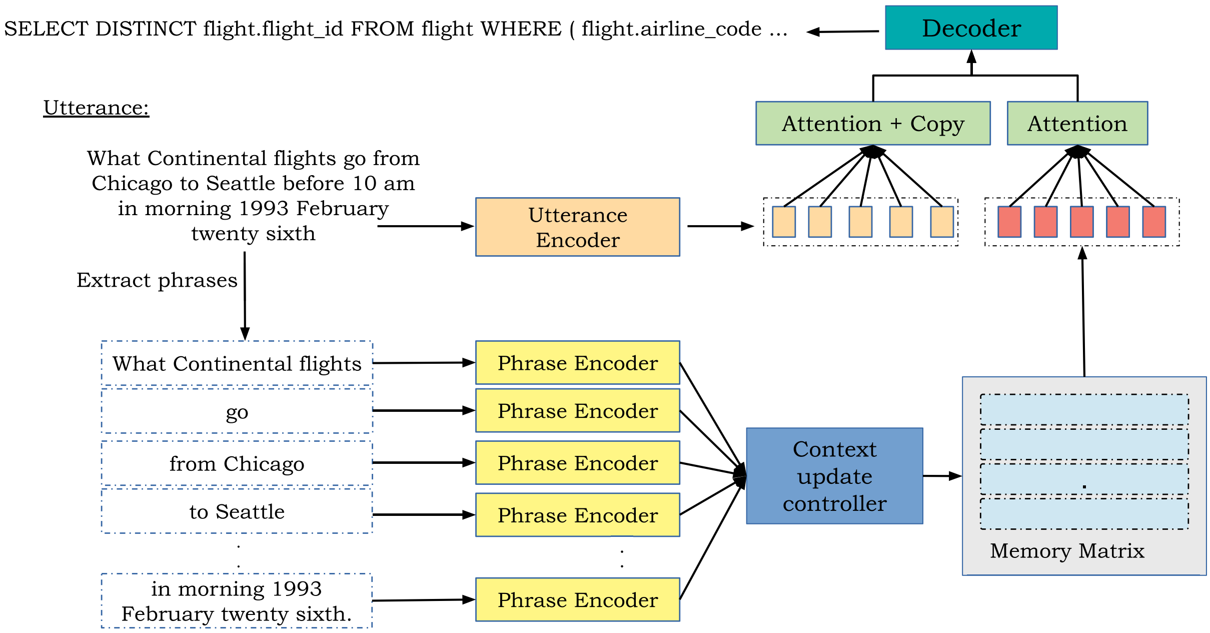

SELECT DISTINCT flight.flight_id FROM flight WHERE (flight.airline_code ...
Decoder
Attention + Copy
Attention
Utterance:
What Continental flights go from Chicago to Seattle before 10 am in morning 1993 February twenty sixth
Utterance Encoder
Extract phrases
What Continental flights
go
from Chicago
to Seattle
in morning 1993 February twenty sixth.
Phrase Encoder
Phrase Encoder
Phrase Encoder
Phrase Encoder
Phrase Encoder
Context update controller
Memory Matrix

Figure 2: Overview of model architecture. Utterances are broken down into segments. Each segment is encoded with the same encoder (same weights) and is processed independently. The context update controller learns to manipulate the memory such that correct discourse information is retained.

3.1 Input Encoder

Each input utterance $X_i = (x_{i,1} \dots x_{i,|X_i|})$ is encoded using a bi-directional LSTM (Hochreiter and Schmidhuber, 1997),

$$h^U_{i,j} = \text{biLSTM}^U(e_{i,j}; h^U_{i,j-1}) \qquad (1)$$

where, $e_{i,j} = \phi(x_{i,j})$ is a learned embedding corresponding to input token $x_{i,j}$ and $h^U_{i,j}$ is the concatenation of the forward and backward LSTM hidden representations at step j. As mentioned earlier, X_i is also segmented into a sequence of phrases denoted as $X_i = (p^1_i \dots p^K_i)$, where K is the number of phrases for utterance X_i. We provide details on how utterances are segmented into phrases in Section 4. For now, suffice it to say that they are obtained from the output of a chunker with some minimal postprocessing (e.g., to merge postmodifiers with NPs or VPs). Each phrase consists of tokens $p^k_i = (x_{i,[s_k:s_k+|p^k_i|]})$, such that $k \in [1, K]$ and $s_k = \sum_{z=1}^{k-1} |p^z_i|$. Each phrase p^k_i is separately encoded using a bi-directional LSTM,

$$h^P_{i,k,j} = \text{biLSTM}^P(e_{i,j}; h^P_{i,k,j-1}) \qquad (2)$$

such that $j \in [s_k : s_k + |p^k_i|]$. As shown in Figure 2, every phrase p^k_i in utterance i is separately encoded using biLSTM^P to obtain a phrase representation $h^P_{i,k}$ by concatenating the final forward and backward hidden representations.

3.2 Context Memory

Our context memory is a matrix $M_i \in \mathbb{R}^{L \times d}$ with L memory slots, each of dimension d, where i is the state of the memory matrix at the i^{th} interaction turn. The goal of context memory is to maintain relevant information required to parse the input utterance at each turn. As shown in Figure 2, this is achieved by learning a *context update controller*, which is responsible for updating the memory at each turn.

For each phrase p^k_i belonging to a sequence of phrases within utterance X_i, the controller decides whether it contains old information that conflicts with information present in the memory or new information that has to be added to the current context. When novel information is introduced, the controller should add it to an empty or least-used memory slot, otherwise the conflicting memory slot should be updated with the latest information. Let t denote the memory update time step such that $t \in [1, n]$, where n is the total number of phrases in interaction I. We simplify notation, using h^P_t instead of $h^P_{i,k}$, to represent the hidden representation of a phrase at time t.

Detecting Conflicts Given phrase representation h^P_t (see Equation (2)), we use a similarity module to detect conflicts between h_t^P and every memory slot in $M_i(m)$ where $m \in [1, L]$; $M_i(m)$ is the m^{th} row representing a memory slot in the memory matrix. Intuitively, low similarity represents new information. Our similarity module is based on a Siamese network architecture (Bromley et al., 1994) that takes phrase hidden representation h_t^P and memory slot $M_i(m)$ and computes a low-dimensional representation using the same neural network weights. The resulting low-dimensional representations are then compared using the cosine distance metric:

$$\hat{w}_c^{t,m} = \frac{\text{sia}(h_t^P) \cdot \text{sia}(M_i(m))}{\max(\| \text{sia}(h_t^P) \|_2 \cdot \| \text{sia}(M_i(m)) \|_2, \epsilon)} \quad (3)$$

where ϵ is a small value for numerical stability and sia is a multi-layer feed-forward network with a $\tanh$ activation function. For hidden representation h, sia is computed as:

$$\hat{h} = W(\tanh(W^l h + b^l) + b) \quad (4)$$

where l represents the layer number and W^l, b^l, W, and b are learnable parameters. We use $\hat{w}_c^{t,m}$ to obtain a similarity distribution w_s^t for updating step t over memory slots. w_s^t represents the probability of dissimilarity (or conflict) which is calculated by computing softmax over cosine similarities with every memory slot $m \in [1..L]$:

$$w_s^t = \text{softmax}([\hat{w}_c^{t,1}; \hat{w}_c^{t,2} \ldots; \hat{w}_c^{t,L}]) \quad (5)$$

We compute softmax over cosine values so that the linear combination of w_s^t with least used weights w_{lu}^t (described below in the memory update paragraph) still represents the probability of update across each memory slot.

Adding New Information To add new information to the memory (i.e., when there is no conflict with any locations), we need to ascertain which memory locations are either empty or rarely used. When the memory is full (i.e., all memory slots are used during previous updates), we update the slot which was least used. This is accomplished by maintaining memory usage weights $w_u^t \in \mathbb{R}^L$ at each update t; w_u^t is initialized with zeros at $t = 0$ and is updated by combining previous memory usage weights w_u^{t-1} with current write weight w_w^t using a decay parameter λ:

$$w_u^t = w_w^t + \lambda w_u^{t-1} \quad (6)$$

where write weights w_w^t are used to compute the write location and are described in the memory update paragraph below. The least used weight vector, w_{lu}^t, at update step t is then calculated as:

$$w_{lu}^t = \text{softmin}(w_u^{t-1}) \quad (7)$$

where for vector x we calculate $\text{softmin}(x) = \exp(-x)/\sum_j \exp(-x_j)$. Hard updates (i.e., using smallest instead of softmin) are also possible. However, we found softmin to be more stable during learning.

Memory Update We wish to compute write location w_w^t given least used weight vector w_{lu}^t and conflict probability distribution w_s^t. Notice that w_s^t and w_{lu}^t are essentially two probability distributions each representing a candidate write location in memory. We learn a convex combination parameter μ that depends on w_s^t,

$$\mu = \sigma(W_\sigma w_s^t + b_\sigma) \quad (8)$$

$$w_w^t = \text{softmax}((\mu w_s^t + (1 - \mu) w_{lu}^t)/\tau) \quad (9)$$

where temperature hyperparameter τ is used to peak the write location. Finally, the memory is updated with current phrase representation h_t^P as

$$M_i^t(m) = M_i^{t-1}(m) + w_w^t(m) h_t^P, \forall m \in [1, L] \quad (10)$$

3.3 Decoder

The output query is generated with an LSTM decoder. As shown in Figure 2, the decoder depends on the memory and utterance representations computed using Equations (10) and (1), respectively. The decoder state at time step s is computed as:

$$h_s^D = \text{LSTM}([\phi^o(y_{i,s-1}); c_{s-1}^M; c_{s-1}^U]; h_{s-1}^D) \quad (11)$$

where ϕ^o is a learned embedding function for output tokens, c_s^U is an utterance context vector, c_{s-1}^M is a memory context vector, and h_{s-1}^D is the previous decoder hidden state. c_s^U is calculated as the weighted sum of all hidden states, where α_s^U is the utterance state attention score:

$$v_s(j) = h_{i,j}^U W^A h_s^D \quad (12)$$

$$\alpha_s^U = \text{softmax}(v_s) \quad (13)$$

$$c_s^U = \sum_j h_{i,j}^U \alpha_s^U(j) \quad (14)$$

Memory state attention score α_s^M and memory context vector c_s^M are computed in a similar manner using memory slots as hidden states.[1] The probability of output query tokens is computed as:

$$P(\hat{w}_{i,s}|X_i, Y_i, I[:i-1]) \propto \\ \exp(\tanh([h_s^D; c_s^U; c_s^M]W^{\hat{o}})W^o + b^o)$$

We further modify the decoder in order to deal with the large number of database values (e.g., city names) common in text-to-SQL semantic parsing tasks. As described in Suhr et al. (2018), we add anonymized token attention scores in the output vocabulary distribution, which enables copying anonymized tokens mentioned in input utterances. The final probability distribution over output vocabulary tokens and anonymized tokens is:

$$P(w_{i,s}) = \text{softmax}(P(\hat{w}_{i,s}) \oplus P(\hat{a}_{i,s})) \quad (15)$$

where $\oplus$ represents concatenation and $P(\hat{a}_{i,s})$ are anonymized token attention scores in the attention distribution α_s^U.

3.4 Training

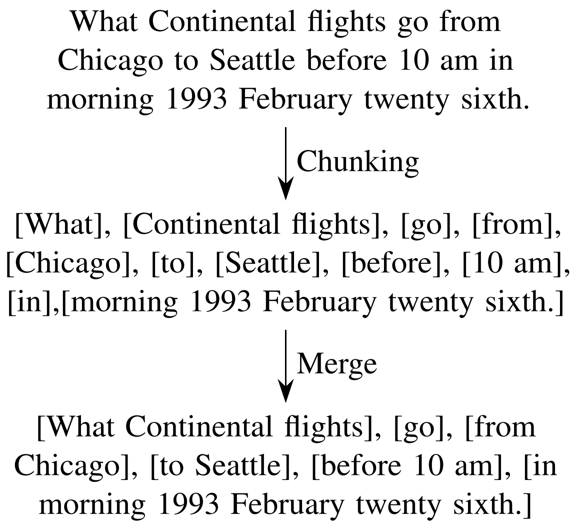
Our model is trained in an end-to-end fashion using a cross-entropy loss. Given a training set of N interactions $\{I^{(l)}\}_{l=1}^N$, such that each interaction $I^{(l)}$ consists of utterances $X_i^{(l)} = (x_{i,1}^{(l)} \dots x_{i,|X_i|^{(l)}})$ paired with output queries $Y_i^{(l)} = (y_{i,1}^{(l)} \dots y_{i,|Y_i|}^{(l)})$, we minimize token cross-entropy loss as:

$$\mathcal{L}(\hat{y}_{i,k}^{(l)}) = -logP\big(\hat{y}_{i,k}^{(l)}|x_i^{(l)}, y_{i,k}^{(l)}, I[:i-1]\big) \quad (16)$$

where, $\hat{y}_{i,k}^{(l)}$ denotes the predicted output token and k is the gold output token index. The total loss is the average of the utterance level losses used for back-propagation.

4 Experimental Setup

We evaluated MemCE, our memory-based context model, on various settings by integrating it with multiple open-source models. We achieve this by replacing the discourse component of related models with MemCE subject to minor or no additional changes. All base models in our experiments use a turn-level hierarchical encoder to capture previous language context. For primary evaluation, we use the ATIS (Hemphill et al., 1990; Dahl et al., 1994) dataset but also present results on SParC (Yu et al., 2019b) and CoSQL (Yu et al., 2019a).



What Continental flights go from Chicago to Seattle before 10 am in morning 1993 February twenty sixth.

Chunking

[What], [Continental flights], [go], [from], [Chicago], [to], [Seattle], [before], [10 am], [in],[morning 1993 February twenty sixth.]

Merge

[What Continental flights], [go], [from Chicago], [to Seattle], [before 10 am], [in morning 1993 February twenty sixth.]

Figure 3: Example of sentence segmentation using chunking and rule-based merging.

Utterance Segmentation We segment each input utterance into a sequence of phrases with a pretrained chunker and then apply a simple rule-based merging procedure to create bigger chunks as an approximation to propositions (Kintsch and van Dijk, 1978). Figure 3 illustrates the process. We used the Flair chunker (Akbik et al., 2018) trained on CONLL-2000 (Tjong Kim Sang and Buchholz, 2000) to identify NP and VP phrases without postmodifiers. Small chunks (e.g., *from, before* in the figure) were subsequently merged into segments using the following rules and NLTK's (Bird et al., 2009) tag-based regex merge:

R1: $left = \langle VP.* \rangle, right = \langle VP.* \rangle$
R2: $left = \langle PP.* \rangle | \langle NP.* \rangle, right = \langle NP \rangle +$
R3: $left = \langle NP.* \rangle, right = \langle VB.* \rangle$
R4: $left = \langle AD.* \rangle, right = \langle NP.* \rangle$

The rules above are applied in order. For each rule we find any chunk whose end matches the left pattern followed by a chunk whose beginning matches the right pattern. Chunks that satisfy this criterion are merged.

We segment utterances and anonymize entities independently and then match entities within segments deterministically. This step is necessary to robustly perform anonymization as in some rare

[1] In experiments we found that using the (raw) memory directly is empirically better to encoding it with an LSTM.

cases, the chunking process will separate entities in two different phrases (e.g., *in Long Beach California that* is chunked as *in Long Beach* and *California that*). This is easily handled by a simple token number matching procedure between the anonymized utterance and corresponding phrases.

Model Configuration Our model is implemented in PyTorch (Paszke et al., 2019). For all experiments, we used the ADAM optimizer (Kingma and Ba, 2015) to minimize the loss function and the initial learning rate was set to 0.001. During training, we used the ReduceLROnPlateau learning rate scheduling strategy on the validation loss, with a decay rate of 0.8. We also applied dropout with 0.5 probability. Dimensions for the word embeddings were set to 300. Following previous work (Zhang et al., 2019) we use pretrained GloVe (Pennington et al., 2014) embeddings for our main experiments on the SparC and CoSQL datasets. For ATIS, word embeddings were not pretrained (Suhr et al., 2018; Zhang et al., 2019). Memory length was chosen as a hyperparameter from the range $[15, 25]$ and the temperature parameter was chosen from $\{0.01, 0.1\}$. Best memory length values for ATIS, SparC, and CoSQL were 25, 16, and 20, respectively. The RNN decoder is a two-layer LSTM and the encoder is a single layer LSTM. The Siamese network in the module which detects conflicting slots uses two hidden layers.

5 Results

In this section, we assess the effectiveness of the MemCE encoder at handling contextual information. We present our results, evaluation methodology, and comparisons against the state of the art.

5.1 Evaluation on ATIS

We primarily focus on ATIS because it contains relatively long interactions (average length is 7) compared with other datasets (e.g., the average length in SParC is 3). Longer interactions present multiple challenges that require non-trivial processing of context, some of which are discussed in Section 6. We use the ATIS dataset split created by Suhr et al. (2018). It contains 27 tables and 162K entries with 1,148/380/130 train/dev/test interactions. The semantic representations are in SQL.

Following Suhr et al. (2018), we measure *query accuracy*, *strict denotation accuracy*, and *relaxed denotation accuracy*. Query accuracy is the percentage of predicted queries that match the reference query. Strict denotation accuracy is the percentage of predicted queries that when executed produce the same results as the reference query. Relaxed accuracy also gives credit to a prediction query that fails to execute if the reference table is empty. In cases where the utterance is ambiguous and there are multiple gold queries, the query or table is considered correct if they match any of the gold labels. We evaluate on both development and test set, and select the best model during training via a separate validation set consisting of 5% of the training data.

Table 1 presents a summary of our results. We compare our approach against a simple Seq2Seq model which is a baseline encoder-decoder without any access to contextual information. Seq2Seq+Concat is a strong baseline which consists of an encoder-decoder model with attention on the current and the *previous three concatenated* utterances. We also compare against the models of Suhr et al. (2018) and Zhang et al. (2019). The former uses a turn-level encoder on top of an utterance-level encoder in a *hierarchical* fashion together with a decoder which learns to copy complete SQL segments from the previous query (SQL segments between consecutive queries are aligned during training using a rule-based procedure). The latter enhances the turn-level encoder by employing an attention mechanism across different turns and additionally introduces a *query editing* mechanism which decides at each decoding step whether to copy from the previous query or insert a new token. Column Enc-Dec in Table 1 describes the various models in terms of the type of encoder/decoder used. LSTM is a vanilla encoder or decoder, HE is a turn-level hierarchical encoder, and Mem is the proposed memory-based encoder. SnipCopy and EditBased respectively refer to Suhr et al.'s 2018 and Zhang et al.'s 2019 decoders. We present two instantiations of our MemCE model with a simple LSTM decoder (Mem-LSTM) and SnipCopy (Mem-SnipCopy). For the sake of completeness, Table 1 also reports the results from Lin et al. (2019), who apply a grammar-based decoder to this task; they also incorporate the interaction history by concatenating the current utterance with the previous three utterances which are encoded with a bi-directional

Model	Enc-Dec	Dev Set			Test Set		
			Denotation			Denotation	
		Query	Relaxed	Strict	Query	Relaxed	Strict
Seq2Seq	LSTM-LSTM	28.7	48.8	43.2	35.7	56.4	53.8
Seq2Seq+Concat	LSTM-LSTM	35.1	59.4	56.7	42.2	66.6	65.8
Suhr et al. (2018)	HE-LSTM	36.0	59.5	58.3	—	—	—
Suhr et al. (2018)	HE-SnipCopy	37.5	63.0	62.5	43.6	69.3	69.2
Zhang et al. (2019)	HE-EditBased	36.2	60.5	60.0	43.9	68.5	68.1
Lin et al. (2019)	LSTM-Grammar	39.1	—	65.8	44.1	—	73.7
MemCE	Mem-LSTM	40.2	63.6	61.2	47.0	70.1	68.9
MemCE	Mem-SnipCopy	39.1	65.5	65.2	45.3	70.2	69.8

Table 1: Model accuracy on the ATIS dataset. HE is a hierarchical interaction encoder, while Mem is the proposed memory-based encoder. LSTM are vanilla encoder/decoder models, while SnipCopy copies SQL segments from the previous query and EditBased adopts a query editing mechanism.

Model	Enc-Dec	CoSQL(D)		CoSQL(T)		SparC(D)		SparC(T)		SparC-DI(T)	
		Q	I	Q	I	Q	I	Q	I	Q	I
CDS2S	HE-LSTM	13.8	2.1	13.9	2.6	21.9	8.1	23.2	7.5	39.5	20.1
CDS2S	HE-SnipCopy	12.3	2.1	—	—	21.7	9.5	20.3	8.1	38.7	24
Liu et al. (2020)	HE-Grammar	33.5	9.6	—	—	41.8	20.6	—	—	57.1	35.3
MemCE+CDS2S	Mem-LSTM	13.4	3.4	—	—	21.2	8.8	—	—	41.3	22.9
MemCE+CDS2S	Mem-SnipCopy	13.1	2.7	—	—	21.4	10.9	—	—	41.5	26.7
MemCE+Liu et al. (2020)	Mem-Grammar	32.8	10.6	28.4	6.2	42.4	21.1	40.3	16.7	55.7	36.3

Table 2: Query (Q) and Interaction (I) accuracy for SParC and CoSQL. We report results on the development (D) and test (T) sets. Sparc-DI is our domain-independent split of SparC. HE is a hierarchical encoder and Mem is the proposed memory-based context encoder. LSTM is a vanilla decoder, SnipCopy copies SQL segments from the previous query, and Grammar refers to a decoder which outputs a sequence of grammar rules rather than tokens. Table cells are filled with—whenever results are not available.

LSTM. All models in Table 1 use entity anonymization; Lin et al. (2019) additionally use identifier linking, namely, string matching heuristic rules to link words or phrases in the input utterance to identifiers in the database (e.g., `city_name_string -> ''BOSTON''`).

As shown in Table 1, MemCE is able to outperform comparison systems. We observe a boost in denotation accuracy when using the SnipCopy decoder instead of an LSTM-based one, although, exact match does not improve. This is possibly because SnipCopy makes it easier to generate long SQL queries by copying segments, but at the same time it suffers from spurious generation and error propagation.

Table 3 presents various ablation studies which evaluate the contribution of individual model components. We use Mem-SnipCopy as our base model and report performance on the ATIS development set following the configuration described in Section 4. We first remove the proposed memory controller described in Section 3.2 and simplify Equation (9) using key-value based attention to calculate w_w^t as

$$\alpha_j = M_i^{t-1}(j) W^P h_t^P \tag{17}$$

$$w_w^t = \mathrm{softmax}(\alpha) \tag{18}$$

We observe a decrease in performance (see second row in Table 3), indicating that the proposed memory controller is helpful in maintaining interaction context.

We performed two ablation experiments to evaluate the usefulness of utterance segmentation. Firstly, instead of the phrases extracted from our segmentation procedure, we employ a variant of our model which operates over individual tokens (see row ‘‘phrases are utterance tokens’’ in Table 3). As can be seen, this strategy is not optimal as results decrease across metrics. We believe

		Denotation	
	Query	Relaxed	Strict
MemCE+SnipCopy	39.1	65.5	65.2
Without memory controller	34.3	58.7	58.1
Phrases are utterance tokens	37.2	61.9	61.7
Phrases are full utterances	36.8	64.2	63.9

Table 3: Ablation results with SnipCopy decoder on the ATIS development set.

operating directly on tokens can lead to ambiguity during update. For example, when processing current phrase *to Boston* given previous utterance *What Continental flights go from Chicago to Seattle*, it is not obvious whether *Boston* should update *Chicago* or *Seattle*. Secondly, we do not use any segmentation at all, not even at the token level. Instead, we treat the entire utterance as a single phrase (see row ‘‘phrases are full utterances’’ in Table 3). If memory's only function is to simply store utterance encodings, then this model becomes comparable to a hierarchical encoder with attention. Again, we observe that performance decreases, which indicates that our system benefits from utterance segmentation. Overall, the ablation studies in Table 3 show that segmentation and its granularity matters. Our heuristic procedure works well for the task at hand, although a learning-based method would be more flexible and potentially lead to further improvements. However, we leave this to future work.

5.2 Evaluation on SParC and CoSQL

In this section we describe our results on SParC and CoSQL. Both datasets assume a cross-domain semantic parsing task in context with SQL as the meaning representation. In addition, for ambiguous utterances, (which cannot be uniquely mapped to SQL given past context), CoSQL also includes clarification questions (and answers). We do not tackle these explicitly but consider them part of the utterance preceding them (e.g., *please list the singers | did you mean list their names? | yes*). Since our primary objective is to study and measure context-dependent language understanding, we created a split of SParC that is denoted as SParC-DI[2], where domains are all seen in training, development, *and* test set. In this way we ensure that no model has the added advantage of being able to handle cross-domain instances while lacking context-dependent language understanding. Table 4 shows the statistics of our SParC-DI split, following a ratio of 80/10/10 percent for the training/development/test set.

	Train	Dev	Test
#Interactions	2869	290	290
#Utterances	8535	851	821

Table 4: Statistics for SParC-DI domain-independent split which has 157 domains in total.

We evaluate model output using exact set match accuracy (Yu et al., 2019b).[3] We report two metrics: *question accuracy*, which is the accuracy considering all utterances independently, and *interaction accuracy*, which is the correct interaction accuracy averaged across interactions. An interaction is marked as correct if all utterances in that interaction are correct. Because utterances in an interaction can be semantically complete (i.e., independent of context), we prefer interaction accuracy.

Table 2 summarizes our results. CDS2S is the context-dependent cross-domain parsing model of Zhang et al. (2019). It is is adapted from Suhr et al. (2018) to include a schema encoder, which is necessary for SparC and CoSQL. It also uses a turn-level hierarchical encoder to represent the interaction history. We also report model variants where the CDS2S encoder is combined with an LSTM-based encoder, SnipCopy (Suhr et al., 2018), and a grammar-based decoder (Liu et al., 2020). The latter decodes SQL queries as a sequence of grammar rules, rather than tokens. We compare the above systems with three variants of our MemCE model that differ in their use of an LSTM decoder, SnipCopy, and the grammar-based decoder of Liu et al. (2020).

Across models and datasets we observe that MemCE improves performance, which suggests that it better captures contextual information as an independent language modeling component. We observe that benefits from our memory-based encoder persist across domains and data splits even

[2] We only considered training and development instances as the test set is not publicly available.

[3] Predicted queries are decomposed into different SQL clauses and scores are computed for each clause separately.

	MemCE		Suhr et al. (2018)	
	Denotation	Query	Denotation	Query
Focus Shift	80.4	50.0	76.7	44.6
Referring Exp	80.0	40.0	70.0	20.0
Ellipsis	69.4	33.3	66.6	25.0
Independent	81.4	61.1	81.3	62.7

Table 5: Model accuracy on specific phenomena (20 interactions, ATIS dev set).

when sophisticated strategies like grammar-based decoding are adopted.

6 Analysis

In this section, we analyze our model's ability to handle important discourse phenomena such as focus shift, referring expressions, and ellipsis. We also showcase its interpretability by examining the behavior of the (learned) memory controller.

6.1 Focus Shift

Our linguistic analysis took place on 20 interactions[4] randomly sampled from the ATIS development set (134 utterances in total). Table 5 shows overall performance statistics for MemCE (MemLSTM) and Suhr et al. (2018) (HE-SnipCopy) on our sample. We annotated the focus of attention in each utterance (underlined in the example below) which we operationalized as the most salient entity (e.g., city) within the utterance (Grosz et al., 1995). Focus shift occurs when the attention transitions from one entity to another. In the interaction below the focus shifts from *flights* in Q2 to *cities* in Q3.

Q1: What flights are provided by American airlines
Q2: What flights are provided by Delta airlines
Q3: Which cities are serviced by both American and Delta airlines

Handling focus shift has been problematic in the context of semantic parsing (Suhr et al., 2018). In our sample, 41.8% of utterances displayed focus shift. Our model was able to correctly parse all utterances in the interaction above and is more apt at handling focus shifts compared to related systems (Suhr et al., 2018). Table 5 reports denotation and query accuracy on our analysis sample.

6.2 Referring Expressions and Ellipsis

Ellipsis refers to the omission of information from an utterance that can be recovered from the context. In the interaction below, Q2 and Q3 exemplify nominal ellipsis, the NP *all flights from Long Beach to Memphis* is elided and ideally should be recovered from the discourse, in order to generate correct SQL queries. Q4 is an example of coreference, *they* refers to the answer of Q3. However, it can also be recovered by considering all previous utterances (i.e., Where do they [flights from Long Beach to Memphis; any day] stop). Because our model explicitly stores information in context, it is able to parse utterances like Q2 and Q4 correctly.

Q1: Please give me all flights from Long Beach to Memphis
Q2: What about 1993 June thirtieth
Q3: How about any day
Q4: Where do they stop

In our ATIS sample, 26.8% of the utterances exhibited ellipsis and 7.5% contained referring expressions. Results in Table 5 show that MemCE is able to better handle both such cases.

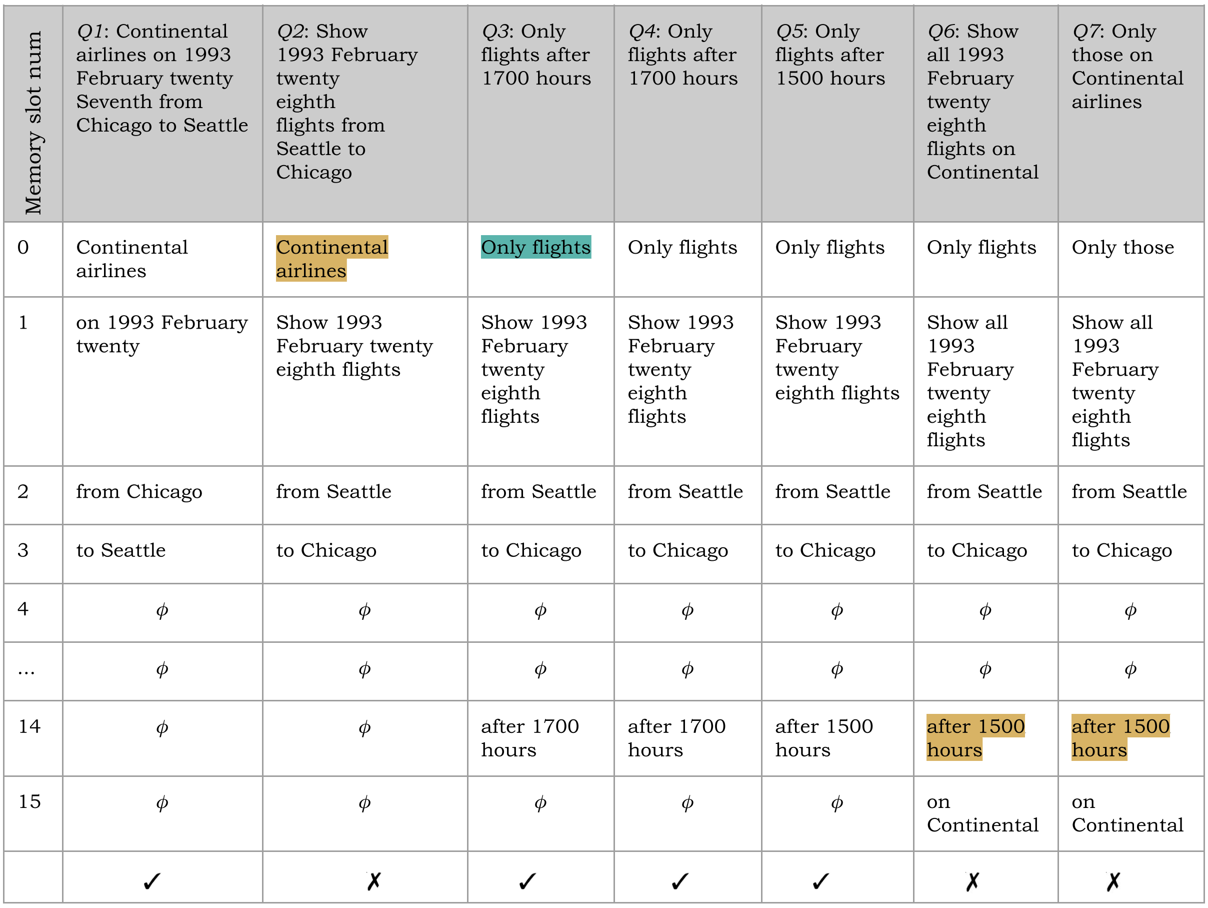
6.3 Memory Interpretation

In this section we delve into the memory controller with the aim of understanding what kind of patterns it learns and where it fails. In Figure 4, we visualize the content of memory for an interaction (top row) from the ATIS development set consisting of seven utterances.[5] Each column in Figure 4 shows the content of memory after processing the corresponding utterance in the interaction. The bottom row indicates whether the final output was correct (✓) or not (✗). For the purpose of clear visualization we took the $\max$ instead of $\mathrm{softmax}$ in Equation (8) to obtain the memory state at any time step.

Q2 presents an interesting case for our model, it is not obvious whether *Continental airlines* from Q1 should be carried forward while processing Q2. The latter is genuinely ambiguous, it could be referring to Continental airlines flights or to flights by any carrier leaving from Seattle to Chicago. If we assume the second interpretation, then Q2 is more or less semantically complete and independent of Q1. Forty-four percent of utterances in our ATIS sample are semantically complete. Although we do not explicitly handle such utterances, our model is able to parse many

[4] Interactions with less than two utterances were discarded.

[5] Q4 was repeated in the dataset. We do the same to maintain consistency and to observe the effect of repetition.



Memory slot num	*Q1*: Continental airlines on 1993 February twenty Seventh from Chicago to Seattle	*Q2*: Show 1993 February twenty eighth flights from Seattle to Chicago	*Q3*: Only flights after 1700 hours	*Q4*: Only flights after 1700 hours	*Q5*: Only flights after 1500 hours	*Q6*: Show all 1993 February twenty eighth flights on Continental	*Q7*: Only those on Continental airlines
0	Continental airlines	Continental airlines	Only flights	Only flights	Only flights	Only flights	Only those
1	on 1993 February twenty	Show 1993 February twenty eighth flights	Show 1993 February twenty eighth flights	Show 1993 February twenty eighth flights	Show 1993 February twenty eighth flights	Show all 1993 February twenty eighth flights	Show all 1993 February twenty eighth flights
2	from Chicago	from Seattle	from Seattle	from Seattle	from Seattle	from Seattle	from Seattle
3	to Seattle	to Chicago	to Chicago	to Chicago	to Chicago	to Chicago	to Chicago
4	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
...	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
14	ϕ	ϕ	after 1700 hours	after 1700 hours	after 1500 hours	after 1500 hours	after 1500 hours
15	ϕ	ϕ	ϕ	ϕ	ϕ	on Continental	on Continental
	✓	✗	✓	✓	✓	✗	✗

Figure 4: Visualization of memory matrix. Rows represent memory content and columns represents the utterance time step. The top row shows the utterances being processed. Each row is marked with a memory slot number which represents the content of memory in that slot. Empty slots are marked with ϕ. The bottom row shows whether the utterance was parsed correctly(✓) or not(✗). 🟨: Stale content in memory with respect to the current utterance. 🟩: Incorrect substitution.

of them correctly because they usually repeat the information mentioned in previous discourse as a single query (see Table 5). Q2 also shows that the memory controller is able to learn the similarity between long phrases: *on 1993 February twenty Seventh* ⇔ *Show 1993 February twenty eighth flights*. It also demonstrates a degree of semantic understanding—that is, it replaces *from Chicago* with *from Seattle* in order to process utterance Q2, rather than simply relying on entity matching.

Figure 4 further shows the kind of mistakes the controller makes which are mostly due to stale content in memory. In utterance Q6 the memory carries over the constraint *after 1500 hours* from the previous utterance, which is not valid since Q6 explicitly states *Show all . . . flights on Continental*. At the same time constraints *from Seattle* and *to Chicago* should carry forward. Knowing which content to keep or discard makes the task challenging.

Another cause of errors relates to reinstating previously nullified constraints. In the interaction below, Q3 reinstates *from Seattle to Chicago*, the focus shifts from flights in Q1 to ground transportation in Q2 and then again to flights in Q3.

Q1: Show flights from Seattle to Chicago
Q2: What ground transportation is available in Chicago
Q3: Show flights after 1500 hours

Handling these issues altogether necessitates a non-trivial way of managing context. Given that our model is trained in an end-to-end fashion, it is encouraging to observe a one-to-one correspondence between memory and the final output which supports our hypothesis that explicitly modeling language context is helpful.

7 Conclusions

In this paper, we presented a memory-based model for context-dependent semantic parsing and evaluated its performance on a text-to-SQL task. Analysis of model output revealed that our approach is able to handle several discourse related phenomena to a large extent. We also analyzed the behavior of the memory controller and observed that it correlates with the model's output decisions. Our study indicates that explicitly modeling context can be helpful for contextual language processing tasks. Our model manipulates information at the phrase level which can be too rigid for fine-grained updates. In the future, we would like to experiment with learning the right level of utterance segmentation for context modeling as well as learning when to reinstate a constraint.

Acknowledgment

We thank Mike Lewis, Miguel Ballesteros, and our anonymous reviewers for their feedback. We are grateful to Alex Lascarides and Ivan Titov for their comments on the paper. This work was supported in part by Huawei and the UKRI Centre for Doctoral Training in Natural Language Processing (grant EP/S022481/1). Lapata acknowledges the support of the European Research Council (award number 681760, ``Translating Multiple Modalities into Text'').

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Yoav Artzi and Luke Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 421–432, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Alan D. Baddeley. 2007. *Working Memory, Thought, and Action*, Oxford University Press, Oxford.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*, O'Reilly Media, Inc.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1994. Signature verification using a ``Siamese'' time delay neural network. In *Advances in Neural Information Processing Systems*, volume 6, pages 737–744. Morgan-Kaufmann.

Giovanni Campagna, Rakesh Ramesh, Silei Xu, Michael Fischer, and Monica S. Lam. 2017. Almond: The architecture of an open, crowdsourced, privacy-preserving, programmable virtual assistant. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 341–350, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE.

Joyce Y. Chai and Rong Jin. 2004. Discourse structure for context question answering. In *Proceedings of the Workshop on Pragmatics of Question Answering at HLT-NAACL 2004*, pages 23–30, Boston, Massachusetts, USA. Association for Computational Linguistics.

Kyunghyun Cho, Aaron Courville, and Yoshua Bengio. 2015. Describing multimedia content using attention-based encoder-decoder networks. *IEEE Transactions on Multimedia*, 17(11):1875–1886. https://doi.org/10.1109/TMM.2015.2477044

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8–11, 1994*.

Jeffrey Dalton, Chenyan Xiong, Vaibhav Kumar, and Jamie Callan. 2020. Cast-19: A dataset for conversational information seeking. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, pages 1985–1988, New York, NY, USA. Association for Computing Machinery. https://doi.org/10.1145/3397271.3401206

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany. Association for Computational Linguistics. https://doi.org/10.18653/v1/P16-1004

Kais Dukes. 2014. SemEval-2014 task 6: Supervised semantic parsing of robotic spatial commands. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 45–53, Dublin, Ireland. Association for Computational Linguistics. https://doi.org/10.3115/v1/S14-2006

Yimai Fang and Simone Teufel. 2016. Improving argument overlap for proposition-based summarisation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 479–485, Berlin, Germany. Association for Computational Linguistics. https://doi.org/10.18653/v1/P16-2078

Jianfeng Gao, Michel Galley, and Lihong Li. 2019. Neural approaches to conversational AI. Foundations and Trends®, in Information Retrieval, 13(2–3):127–298. https://doi.org/10.1561/1500000074

Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225. https://doi.org/10.21236/ADA324949

Barbara J. Grosz and Candace L. Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-SQL in cross-domain database with intermediate representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535, Florence, Italy. Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1444

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24–27, 1990*. https://doi.org/10.3115/116580.116613

J. Hobbs. 1985. On the coherence and structure of discourse. *CSLI*, 85(37).

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735, PubMed: 9377276

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

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

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany. Association for Computational Linguistics. https://doi.org/10.18653/v1/P16-1002

Aishwarya Kamath and Rajarshi Das. 2019. A survey on semantic parsing. In *Automated Knowledge Base Construction (AKBC)*.

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

Walter Kintsch and Teun A. van Dijk. 1978. Toward a model of text comprehension and production. *Psychological Review*, 85(5):363–394. https://doi.org/10.1037/0033-295X.85.5.363

Kevin Lin, Ben Bogin, Mark Neumann, Jonathan Berant, and Matt Gardner. 2019. Grammar-based neural text-to-sql generation. *ArXiv*, abs/1905.13326.

Qian Liu, Bei Chen, Jiaqi Guo, Jian-Guang Lou, Bin Zhou, and Dongmei Zhang. 2020. How far are we from effective context modeling? An exploratory study on semantic parsing in context. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3580–3586. International Joint Conferences on Artificial Intelligence Organization. https://doi.org/10.24963/ijcai.2020/495

William C. Mann and Sandra A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-interdisciplinary Journal for the Study of Discourse*, 8(3):243–281. https://doi.org/10.1515/text.1.1988.8.3.243

Scott Miller, David Stallard, Robert Bobrow, and Richard Schwartz. 1996. A fully statistical approach to natural language interfaces. In *34th Annual Meeting of the Association for Computational Linguistics*, pages 55–61, Santa Cruz, California, USA. Association for Computational Linguistics. https://doi.org/10.3115/981863.981871

Fatma Őzcan, Abdul Quamar, Jaydeep Sen, Chuan Lei, and Vasilis Efthymiou. 2020. State of the art and open challenges in natural language interfaces to data. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, pages 2629–2636, New York, NY, USA. Association for Computing Machinery. https://doi.org/10.1145/3318464.3383128

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, pages 8026–8037. Curran Associates, Inc.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1162

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1842–1850, New York, New York, USA. PMLR.

Shashank Srivastava, Amos Azaria, and Tom Mitchell. 2017. Parsing natural language conversations using contextual cues. In *Pro ceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 4089–4095.

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

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, volume 28, pages 2440–2448. Curran Associates, Inc.

Mingyu Sun and Joyce Y. Chai. 2007. Discourse processing for context question answering based on linguistic knowledge. *Knowledge-Based Systems*, 20(6):511–526. Special Issue On Intelligent User Interfaces. https://doi.org/10.1016/j.knosys.2007.04.005

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task chunking. In *Fourth Conference on Computational Natural Language Learning and the Second Learning Language in Logic Workshop*. https://doi.org/10.3115/1117601.1117631

Ellen M. Voorhees. 2004. Overview of TREC 2004. In *Proceedings of the Thirteenth Text REtrieval Conference, TREC 2004, Gaithersburg, Maryland, USA, November 16–19, 2004*, volume 500–261 of *NIST Special Publication*. National Institute of Standards and Technology (NIST).

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.677

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings*.

Semih Yavuz, Izzeddin Gur, Yu Su, and Xifeng Yan. 2018. What it takes to achieve 100% condition accuracy on WikiSQL. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1702–1711, Brussels, Belgium. Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1197

Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. 2018a. SyntaxSQLNet: Syntax tree networks for complex and cross-domain text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1653–1663, Brussels, Belgium. Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1193

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter Lasecki, and Dragomir Radev. 2019a. CoSQL: A conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1962–1979, Hong Kong, China. Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1204

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018b. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1425

Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Tao Chen, Emily Ji, Shreya Dixit, David Proctor, Sungrok Shim, Jonathan Kraft, Vincent Zhang, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019b. SParC: Cross-domain semantic parsing in context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4511–4523, Florence, Italy. Association for Computational Linguistics.

Luke Zettlemoyer and Michael Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proceedings of the*

Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, pages 976–984, Suntec, Singapore. Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1443

Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019. Editing-based SQL query generation for cross-domain context-dependent questions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5338–5349, Hong Kong, China. Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1537

Victor Zhong, Caiming Xiong, and Richard Socher. 1995. Seq2SQL: Generating structured queries from natural language using reinforcement learning.