Generating Logical Forms from Graph Representations of Text and Entities

Peter Shaw[1], **Philip Massey**[1], **Angelica Chen**[1], **Francesco Piccinno**[2], **Yasemin Altun**[2]
[1]Google [2]Google Research
{petershaw,pmassey,angelicachen,piccinno,altun}@google.com

Abstract

Structured information about entities is critical for many semantic parsing tasks. We present an approach that uses a Graph Neural Network (GNN) architecture to incorporate information about relevant entities and their relations during parsing. Combined with a decoder copy mechanism, this approach provides a conceptually simple mechanism to generate logical forms with entities. We demonstrate that this approach is competitive with the state-of-the-art across several tasks without pre-training, and outperforms existing approaches when combined with BERT pre-training.

1 Introduction

Semantic parsing maps natural language utterances into structured meaning representations. The representation languages vary between tasks, but typically provide a precise, machine interpretable logical form suitable for applications such as question answering (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2007; Liang et al., 2013; Berant et al., 2013). The logical forms typically consist of two types of symbols: a vocabulary of operators and domain-specific predicates or functions, and entities grounded to some knowledge base or domain.

Recent approaches to semantic parsing have cast it as a sequence-to-sequence task (Dong and Lapata, 2016; Jia and Liang, 2016; Ling et al., 2016), employing methods similar to those developed for neural machine translation (Bahdanau et al., 2014), with strong results. However, special consideration is typically given to handling of entities. This is important to improve generalization and computational efficiency, as most tasks require handling entities unseen during training, and the set of unique entities can be large.

Some recent approaches have replaced surface forms of entities in the utterance with placeholders (Dong and Lapata, 2016). This requires a preprocessing step to completely disambiguate entities and replace their spans in the utterance. Additionally, for some tasks it may be beneficial to leverage relations between entities, multiple entity candidates per span, or entity candidates without a corresponding span in the utterance, while generating logical forms.

Other approaches identify only types and surface forms of entities while constructing the logical form (Jia and Liang, 2016), using a separate post-processing step to generate the final logical form with grounded entities. This ignores potentially useful knowledge about relevant entities.

Meanwhile, there has been considerable recent interest in Graph Neural Networks (GNNs) (Scarselli et al., 2009; Li et al., 2016; Kipf and Welling, 2017; Gilmer et al., 2017; Veličković et al., 2018) for effectively learning representations for graph structures. We propose a GNN architecture based on extending the self-attention mechanism of the Transformer (Vaswani et al., 2017) to make use of relations between input elements.

We present an application of this GNN architecture to semantic parsing, conditioning on a graph representation of the given natural language utterance and potentially relevant entities. This approach is capable of handling ambiguous and potentially conflicting entity candidates jointly with a natural language utterance, relaxing the need for completely disambiguating a set of linked entities before parsing. This graph formulation also enables us to incorporate knowledge about the relations between entities where available. Combined with a copy mechanism while decoding, this approach also provides a conceptually simple method for generating logical forms with grounded entities.

We demonstrate the capability of the pro-

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 95–106
Florence, Italy, July 28 - August 2, 2019.

Dataset	Example
GEO	**x** : *which states does the mississippi run through ?* **y** : `answer ( state ( traverse_1(` riverid (mississippi))))
ATIS	**x** : *in denver what kind of ground transportation is there from the airport to downtown* **y** : `( _lambda $0 e ( _and ( _ground_transport $0 ) ( _to_city $0` denver : _ci) (`_from_airport $0` den : _ap)))
SPIDER	**x** : *how many games has each stadium held ?* **y** : `SELECT T1 .` id , `count ( * ) FROM` stadium `AS T1 JOIN` game `AS T2 ON T1 .` id `= T2 .` stadium_id `GROUP BY T1 .` id

Table 1: Example input utterances, $\mathbf{x}$, and meaning representations, $\mathbf{y}$, with entities underlined.

posed architecture by achieving competitive results across 3 semantic parsing tasks. Further improvements are possible by incorporating a pre-trained BERT (Devlin et al., 2018) encoder within the architecture.

2 Task Formulation

Our goal is to learn a model for semantic parsing from pairs of natural language utterances and structured meaning representations. Let the natural language utterance be represented as a sequence $\mathbf{x} = (x_1, \ldots, x_{|\mathbf{x}|})$ of $|\mathbf{x}|$ tokens, and the meaning representation be represented as a sequence $\mathbf{y} = (y_1, \ldots, y_{|\mathbf{y}|})$ of $|\mathbf{y}|$ elements.

The goal is to estimate $p(\mathbf{y} \mid \mathbf{x})$, the conditional probability of the meaning representation $\mathbf{y}$ given utterance $\mathbf{x}$, which is augmented by a set of potentially relevant entities.

Input Utterance Each token $x_i \in \mathcal{V}^{in}$ is from a vocabulary of input tokens.

Entity Candidates Given the input utterance $\mathbf{x}$, we retrieve a set, $\mathbf{e} = \{e_1, \ldots, e_{|\mathbf{e}|}\}$, of potentially relevant entity candidates, with $\mathbf{e} \subseteq \mathcal{V}^e$, where $\mathcal{V}^e$ is in the set of all entities for a given domain. We assume the availability of an entity candidate generator for each task to generate $\mathbf{e}$ given $\mathbf{x}$, with details given in § 5.2.

For each entity candidate, $e \in \mathcal{V}^e$, we require a set of task-specific attributes containing one or more elements from $\mathcal{V}^a$. These attributes can be NER types or other characteristics of the entity, such as "city" or "river" for some of the entities listed in Table 1. Whereas $\mathcal{V}^e$ can be quite large for open domains, or even infinite if it includes sets such as the natural numbers, $\mathcal{V}^a$ is typically much smaller. Therefore, we can effectively learn representations for entities given their set of attributes, from our set of example pairs.

Edge Labels In addition to $\mathbf{x}$ and $\mathbf{e}$ for a particular example, we also consider the $(|\mathbf{x}|+|\mathbf{e}|)^2$ pairwise relations between all tokens and entity candidates, represented as edge labels.

The edge label between tokens x_i and x_j corresponds to the relative sequential position, $j - i$, of the tokens, clipped to within some range.

The edge label between token x_i and entity e_j, and vice versa, corresponds to whether x_i is within the span of the entity candidate e_j, or not.

The edge label between entities e_i and e_j captures the relationship between the entities. These edge labels can have domain-specific interpretations, such as relations in a knowledge base, or any other type of entity interaction features. For tasks where this information is not available or useful, a single generic label between entity candidates can be used.

Output We consider the logical form, $\mathbf{y}$, to be a linear sequence (Vinyals et al., 2015b). We tokenize based on the syntax of each domain. Our formulation allows each element of $\mathbf{y}$ to be either an element of the output vocabulary, $\mathcal{V}^{out}$, or an entity copied from the set of entity candidates $\mathbf{e}$. Therefore, $y_i \in \mathcal{V}^{out} \cup \mathcal{V}^e$. Some experiments in §5.2 also allow elements of $\mathbf{y}$ to be tokens $\in \mathcal{V}^{in}$ from $\mathbf{x}$ that are copied from the input.

3 Model Architecture

Our model architecture is based on the Transformer (Vaswani et al., 2017), with the self-attention sub-layer extended to incorporate relations between input elements, and the decoder extended with a copy mechanism.

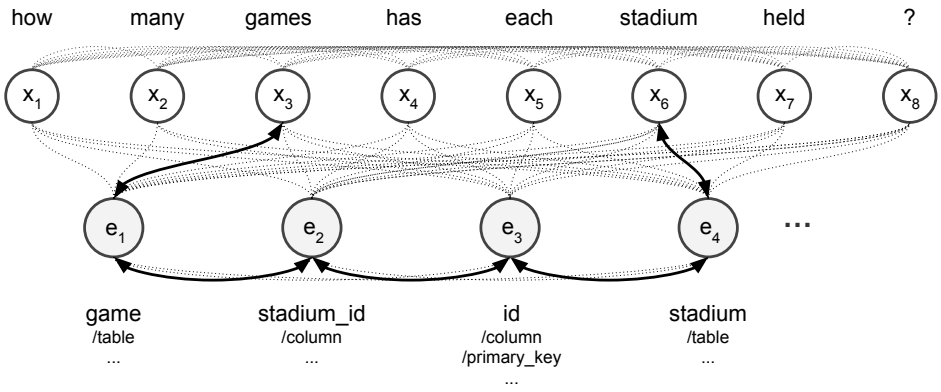

Figure 1: We use an example from SPIDER to illustrate the model inputs: tokens from the given utterance, $\mathbf{x}$, a set of potentially relevant entities, $\mathbf{e}$, and their relations. We selected two edge label types to highlight: edges denoting that an entity spans a token, and edges between entities that, for SPIDER, indicate a foreign key relationship between columns, or an ownership relationship between columns and tables.

3.1 GNN Sub-layer

We extend the Transformer's self-attention mechanism to form a Graph Neural Network (GNN) sub-layer that incorporates a fully connected, directed graph with edge labels.

The sub-layer maps an ordered sequence of node representations, $\mathbf{u} = (u_1, \ldots, u_{|\mathbf{u}|})$, to a new sequence of node representations, $\mathbf{u}' = (u'_1, \ldots, u'_{|\mathbf{u}|})$, where each node is represented $\in \mathbb{R}^d$. We use r_{ij} to denote the edge label corresponding to u_i and u_j.

We implement this sub-layer in terms of a function $f(m, l)$ over a node representation $m \in \mathbb{R}^d$ and an edge label l that computes a vector representation in $\mathbb{R}^{d'}$. We use n_{heads} parallel attention heads, with $d' = d/n_{heads}$. For each head k, the new representation for the node u_i is computed by

$$u_i^{k\prime} = \sum_{j=1}^{|\mathbf{u}|} \alpha_{ij} f(u_j, r_{ij}), \tag{1}$$

where each coefficient α_{ij} is a softmax over the scaled dot products s_{ij},

$$s_{ij} = \frac{(\mathbf{W}^q u_i)^\intercal f(u_j, r_{ij})}{\sqrt{d'}}, \tag{2}$$

and $\mathbf{W}^q$ is a learned matrix. Finally, we concatenate representations from each head,

$$u'_i = \mathbf{W}^h \left[u_i^{1\prime} \mid \cdots \mid u_i^{n_{heads}\prime} \right], \tag{3}$$

where $\mathbf{W}^h$ is another learned matrix and $[\ \cdots\ ]$ denotes concatenation.

If we implement f as,

$$f(m, l) = \mathbf{W}^r m, \tag{4}$$

where $\mathbf{W}^r \in \mathbb{R}^{d' \times d}$ is a learned matrix, then the sub-layer would be effectively identical to self-attention as initially proposed in the Transformer (Vaswani et al., 2017).

We focus on two alternative formulations of f that represent edge labels as learned matrices and learned vectors.

Edge Matrices The first formulation represents edge labels as linear transformations, a common parameterization for GNNs (Li et al., 2016),

$$f(m, l) = \mathbf{W}^l m, \tag{5}$$

where $\mathbf{W}^l \in \mathbb{R}^{d' \times d}$ is a learned embedding matrix per edge label.

Edge Vectors The second formulation represents edge labels as additive vectors using the same formulation as Shaw et al. (2018),

$$f(m, l) = \mathbf{W}^r m + \mathbf{w}^l, \tag{6}$$

where $\mathbf{W}^r \in \mathbb{R}^{d' \times d}$ is a learned matrix shared by all edge labels, and $w^l \in \mathbb{R}^d$ is a learned embedding vector per edge label l.

3.2 Encoder

Input Representations Before the initial encoder layer, tokens are mapped to initial representations using either a learned embedding table for $\mathcal{V}^{in}$, or the output of a pre-trained BERT (Devlin et al., 2018) encoder. Entity candidates are mapped to initial representations using the mean of the embeddings for each of the entity's attributes, based on a learned embedding table for

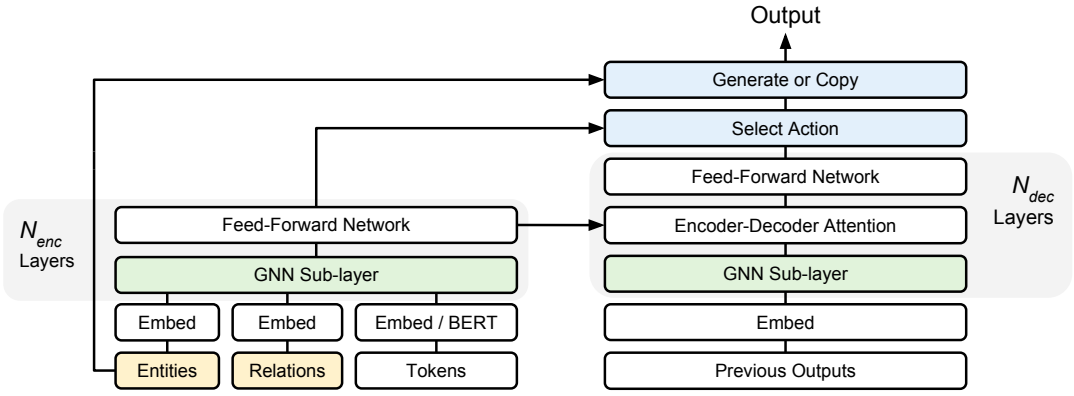

Figure 2: Our model architecture is based on the Transformer (Vaswani et al., 2017), with two modifications. First, the self-attention sub-layer has been extended to be a GNN that incorporates edge representations. In the encoder, the GNN sub-layer is conditioned on tokens, entities, and their relations. Second, the decoder has been extended to include a copy mechanism (Vinyals et al., 2015a). We can optionally incorporate a pre-trained model such as BERT to generate contextual token representations.

$\mathcal{V}^a$. We also concatenate an embedding representing the node type, token or entity, to each input representation.

We assume some arbitrary ordering for entity candidates, generating a combined sequence of initial node representations for tokens and entities. We have edge labels between every pair of nodes as described in § 2.

Encoder Layers Our encoder layers are essentially identical to the Transformer, except with the proposed extension to self-attention to incorporate edge labels. Therefore, each encoder layer consists of two sub-layers. The first is the GNN sub-layer, which yields new sets of token and entity representations. The second sub-layer is an element-wise feed-forward network. Each sub-layer is followed by a residual connection and layer normalization (Ba et al., 2016). We stack N_{enc} encoder layers, yielding a final set of token representations, $\mathbf{w_x}^{(N_{enc})}$, and entity representations, $\mathbf{w_e}^{(N_{enc})}$.

3.3 Decoder

The decoder auto-regressively generates output symbols, $y_1, \ldots, y_{|\mathbf{y}|}$. It is similarly based on the Transformer (Vaswani et al., 2017), with the self-attention sub-layer replaced by the GNN sub-layer. Decoder edge labels are based only on the relative timesteps of the previous outputs. The encoder-decoder attention layer considers both encoder outputs $\mathbf{w_x}^{(N_{enc})}$ and $\mathbf{w_e}^{(N_{enc})}$, jointly normalizing attention weights over tokens and entity candidates. We stack N_{dec} decoder layers to produce an output vector representation at each output step, $z_j \in \mathbb{R}^{d_z}$, for $j \in \{1, \ldots, |\mathbf{y}|\}$.

We allow the decoder to copy tokens or entity candidates from the input, effectively combining a Pointer Network (Vinyals et al., 2015a) with a standard softmax output layer for selecting symbols from an output vocabulary (Gu et al., 2016; Gulcehre et al., 2016; Jia and Liang, 2016). We define a latent action at each output step, a_j for $j \in \{1, \ldots, |\mathbf{y}|\}$, using similar notation as Jia et al. (2016). We normalize action probabilities with a softmax over all possible actions.

Generating Symbols We can generate a symbol, denoted $\texttt{Generate}[i]$,

$$P(a_j = \texttt{Generate}[i] \mid \mathbf{x}, \mathbf{y}_{1:j-1}) \propto \exp(z_j^\intercal w_i^{out}), \quad (7)$$

where w_i^{out} is a learned embedding vector for the element $\in \mathcal{V}^{out}$ with index i. If $a_j = \texttt{Generate}[i]$, then y_j will be the element $\in \mathcal{V}^{out}$ with index i.

Copying Entities We can also copy an entity candidate, denoted $\texttt{CopyEntity}[i]$,

$$P(a_j = \texttt{CopyEntity}[i] \mid \mathbf{x}, \mathbf{y}_{1:j-1}) \propto \exp((z_j \mathbf{W}^e)^\intercal w_{e_i}^{(N_{enc})}), \quad (8)$$

where $\mathbf{W}^e$ is a learned matrix, and $i \in \{1, \ldots, |\mathbf{e}|\}$. If $a_j = \texttt{CopyEntity}[i]$, then $y_j = e_i$.

4 Related Work

Various approaches to learning semantic parsers from pairs of utterances and logical forms have been developed over the years (Tang and Mooney, 2000; Zettlemoyer and Collins, 2007; Kwiatkowski et al., 2011; Andreas et al., 2013). More recently, encoder-decoder architectures have been applied with strong results (Dong and Lapata, 2016; Jia and Liang, 2016).

Even for tasks with relatively small domains of entities, such as GEO and ATIS, it has been shown that some special consideration of entities within an encoder-decoder architecture is important to improve generalization. This has included extending decoders with copy mechanisms (Jia and Liang, 2016) and/or identifying entities in the input as a pre-processing step (Dong and Lapata, 2016).

Other work has considered open domain tasks, such as WEBQUESTIONSSP (Yih et al., 2016). Recent approaches have typically relied on a separate entity linking model, such as S-MART (Yang and Chang, 2015), to provide a single disambiguated set of entities to consider. In principle, a learned entity linker could also serve as an entity candidate generator within our framework, although we do not explore such tasks in this work.

Considerable recent work has focused on constrained decoding of various forms within an encoder-decoder architecture to leverage the known structure of the logical forms. This has led to approaches that leverage this structure during decoding, such as using tree decoders (Dong and Lapata, 2016; Alvarez-Melis and Jaakkola, 2017) or other mechanisms (Dong and Lapata, 2018; Goldman et al., 2017). Other approaches use grammar rules to constrain decoding (Xiao et al., 2016; Yin and Neubig, 2017; Krishnamurthy et al., 2017; Yu et al., 2018b). We leave investigation of such decoder constraints to future work.

Many formulations of Graph Neural Networks (GNNs) that propagate information over local neighborhoods have recently been proposed (Li et al., 2016; Kipf and Welling, 2017; Gilmer et al., 2017; Veličković et al., 2018). Recent work has often focused on large graphs (Hamilton et al., 2017) and effectively propagating information over multiple graph steps (Xu et al., 2018). The graphs we consider are relatively small and are fully-connected, avoiding some of the challenges posed by learning representations for large, sparsely connected graphs.

Other recent work related to ours has considered GNNs for natural language tasks, such as combining structured and unstructured data for question answering (Sun et al., 2018), or for representing dependencies in tasks such as AMR parsing and machine translation (Beck et al., 2018; Bastings et al., 2017). The approach of Krishnamurthy et al. (2017) similarly considers ambiguous entity mentions jointly with query tokens for semantic parsing, although does not directly consider a GNN.

Previous work has interpreted the Transformer's self-attention mechanism as a GNN (Veličković et al., 2018; Battaglia et al., 2018), and extended it to consider relative positions as edge representations (Shaw et al., 2018). Previous work has also similarly represented edge labels as vectors, as opposed to matrices, in order to avoid over-parameterizing the model (Marcheggiani and Titov, 2017).

5 Experiments

5.1 Semantic Parsing Datasets

We consider three semantic parsing datasets, with examples given in Table 1.

GEO The GeoQuery dataset consists of natural language questions about US geography along with corresponding logical forms (Zelle and Mooney, 1996). We follow the convention of Zettlemoyer and Collins (2005) and use 600 training examples and 280 test examples. We use logical forms based on Functional Query Language (FunQL) (Kate et al., 2005).

ATIS The Air Travel Information System (ATIS) dataset consists of natural language queries about travel planning (Dahl et al., 1994). We follow Zettlemoyer and Collins (2007) and use 4473 training examples, 448 test examples, and represent the logical forms as lambda expressions.

SPIDER This is a large-scale text-to-SQL dataset that consists of 10,181 questions and 5,693 unique complex SQL queries across 200 database tables spanning 138 domains (Yu et al., 2018c). We use the standard training set of 8,659 training example and development set of 1,034 examples, split across different tables.

5.2 Experimental Setup

Model Configuration We configured hyperparameters based on performance on the validation set for each task, if provided, otherwise cross-validated on the training set.

For the encoder and decoder, we selected the number of layers from $\{1, 2, 3, 4\}$ and embedding and hidden dimensions from $\{64, 128, 256\}$, setting the feed forward layer hidden dimensions $4\times$ higher. We employed dropout at training time with $P_{dropout}$ selected from $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$. We used 8 attention heads for each task. We used a clipping distance of 8 for relative position representations (Shaw et al., 2018).

We used the Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 10^{-9}$, and tuned the learning rate for each task. We used the same warmup and decay strategy for learning rate as Vaswani et al. (2017), selecting a number of warmup steps up to a maximum of 3000. Early stopping was used to determine the total training steps for each task. We used the final checkpoint for evaluation. We batched training examples together, and selected batch size from $\{32, 64, 128, 256, 512\}$. During training we used masked self-attention (Vaswani et al., 2017) to enable parallel decoding of output sequences. For evaluation, we used greedy search.

We used a simple strategy of splitting each input utterance on spaces to generate a sequence of tokens. We mapped any token that didn't occur at least 2 times in the training dataset to a special out-of-vocabulary token. For experiments that used BERT, we instead used the same wordpiece (Wu et al., 2016) tokenization as used for pre-training.

BERT For some of our experiments, we evaluated incorporating a pre-trained BERT (Devlin et al., 2018) encoder by effectively using the output of the BERT encoder in place of a learned token embedding table. We then continue to use graph encoder and decoder layers with randomly initialized parameters in addition to BERT, so there are many parameters that are not pre-trained. The additional encoder layers are still necessary to condition on entities and relations.

We achieved best results by freezing the pre-trained parameters for an initial number of steps, and then jointly fine-tuning all parameters, similar to existing approaches for gradual unfreezing (Howard and Ruder, 2018). When unfreezing the pre-trained parameters, we restart the learning rate schedule. We found this to perform better than keeping pre-trained parameters either entirely frozen or entirely unfrozen during fine-tuning.

We used $\text{BERT}_{\text{LARGE}}$ (Devlin et al., 2018), which has 24 layers. For fine tuning we used the same Adam optimizer with weight decay and learning rate decay as used for BERT pre-training. We reduced batch sizes to accommodate the significantly larger model size, and tuned learning rate, warm up steps, and number of frozen steps for pre-trained parameters.

Entity Candidate Generator We use an entity candidate generator that, given $\mathbf{x}$, can retrieve a set of potentially relevant entities, $\mathbf{e}$, for the given domain. Although all generators share a common interface, their implementation varies across tasks.

For GEO and ATIS we use a lexicon of entity aliases in the dataset and attempt to match with ngrams in the query. Each entity has a single attribute corresponding to the entity's type. We used binary valued relations between entity candidates based on whether entity candidate spans overlap, but experiments did not show significant improvements from incorporating these relations.

For SPIDER, we generalize our notion of entities to include tables and table columns. We include all relevant tables and columns as entity candidates, but make use of Levenshtein distance between query ngrams and table and column names to determine edges between tokens and entity candidates. We use attributes based on the types and names of tables and columns. Edges between entity candidates capture relations between columns and the table they belong to, and foreign key relations.

For GEO, ATIS, and SPIDER, this leads to 19.5%, 32.7%, and 74.6% of examples containing at least one span associated with multiple entity candidates, respectively, indicating some entity ambiguity.

Further details on how entity candidate generators were constructed are provided in § A.1.

Output Sequences We pre-processed output sequences to identify entity argument values, and replaced those elements with references to entity candidates in the input. In cases where our entity candidate generator did not retrieve an entity that was used as an argument, we dropped the example from the training data set or considered it incorrect

Method	GEO	ATIS
Kwiatkowski et al. (2013)	89.0	—
Liang et al. (2013)	87.9	—
Wang et al. (2014)	—	**91.3**
Zhao and Huang (2015)	88.9	84.2
Jia and Liang (2016)	89.3	83.3
– data augmentation	85.0	76.3
Dong and Lapata (2016) †	87.1	84.6
Rabinovich et al. (2017) †	87.1	85.9
Dong and Lapata (2018) †	88.2	87.7
Ours		
GNN w/ edge matrices	82.5	84.6
GNN w/ edge vectors	89.3	87.1
GNN w/ edge vectors + BERT	**92.5**	89.7

Method	SPIDER
Xu et al. (2017)	10.9
Yu et al. (2018a)	8.0
Yu et al. (2018b)	24.8
– data augmentation	18.9
Ours	
GNN w/ edge matrices	29.3
GNN w/ edge vectors	**32.1**
GNN w/ edge vectors + BERT	23.5

Table 2: We report accuracies on GEO, ATIS, and SPIDER for various implementations of our GNN sub-layer. For GEO and ATIS, we use † to denote neural approaches that disambiguate and replace entities in the utterance as a pre-processing step. For SPIDER, the evaluation set consists of examples for databases unseen during training.

if in the test set.

Evaluation To evaluate accuracy, we use exact match accuracy relative to gold logical forms. For GEO we directly compare output symbols. For ATIS, we compare normalized logical forms using canonical variable naming and sorting for unordered arguments (Jia and Liang, 2016). For SPIDER we use the provided evaluation script, which decomposes each SQL query and conducts set comparison within each clause without values. All accuracies are reported on the test set, except for SPIDER where we report and compare accuracies on the development set.

Copying Tokens To better understand the effect of conditioning on entities and their relations, we also conducted experiments that considered an alternative method for selecting and disambiguating entities similar to Jia et al. (2016). In this approach we use our model's copy mechanism to copy tokens corresponding to the surface forms of entity arguments, rather than copying entities directly.

$$P(a_j = \texttt{CopyToken}[i] \mid \mathbf{x}, \mathbf{y}_{1:j-1}) \propto \exp((z_j \mathbf{W}^x)^\intercal w_{x_i}^{(N_{enc})}), \quad (9)$$

where $\mathbf{W}^x$ is a learned matrix, and where $i \in \{1, \ldots, |\mathbf{x}|\}$ refers to the index of token $x_i \in \mathcal{V}^{in}$. If $a_j = \texttt{CopyToken}[i]$, then $y_j = x_i$.

This allows us to ablate entity information in the input while still generating logical forms. When copying tokens, the decoder determines the type of the entity using an additional output symbol. For GEO, the actual entity can then be identified as a post-processing step, as a type and surface form is sufficient. For other tasks this could require a more complicated post-processing step to disambiguate entities given a surface form and type.

Method	GEO
Copying Entities	
GNN w/ edge vectors + BERT	**92.5**
GNN w/ edge vectors	89.3
Copying Tokens	
GNN w/ edge vectors	87.9
– entity candidates, $\mathbf{e}$	84.3
BERT	89.6

Table 3: Experimental results for copying tokens instead of entities when decoding, with and without conditioning on the set of entity candidates, $\mathbf{e}$.

5.3 Results and Analysis

Accuracies on GEO, ATIS, and SPIDER are shown in Table 2.

GEO and ATIS Without pre-training, and despite adding a bit of entity ambiguity, we achieve similar results to other recent approaches that disambiguate and replace entities in the utterance as a pre-processing step during both training and evaluating (Dong and Lapata, 2016, 2018). When incorporating BERT, we increase absolute accuracies over Dong and Lapata (2018) on GEO and ATIS by 3.2% and 2.0%, respectively. Notably,

they also present techniques and results that leverage constrained decoding, which our approach would also likely further benefit from.

For GEO, we find that when ablating all entity information in our model and copying tokens instead of entities, we achieve similar results as Jia and Liang (2016) when also ablating their data augmentation method, as shown in Table 3. This is expected, since when ablating entities completely, our architecture essentially reduces to the same sequence-to-sequence task setup. These results demonstrate the impact of conditioning on the entity candidates, as it improves performance even on the token copying setup. It appears that leveraging BERT can partly compensate for not conditioning on entity candidates, but combining BERT with our GNN approach and copying entities achieves 2.9% higher accuracy than using only a BERT encoder and copying tokens.

For ATIS, our results are outperformed by Wang et al. (2014) by 1.6%. Their approach uses hand-engineered templates to build a CCG lexicon. Some of these templates attempt to handle the specific types of ungrammatical utterances in the ATIS task.

SPIDER For SPIDER, a relatively new dataset, there is less prior work. Competitive approaches have been specific to the text-to-SQL task (Xu et al., 2017; Yu et al., 2018a,b), incorporating task-specific methods to condition on table and column information, and incorporating SQL-specific structure when decoding. Our approach improves absolute accuracy by +7.3% relative to Yu et al. (2018b) without using any pre-trained language representations, or constrained decoding. Our approach could also likely benefit from some of the other aspects of Yu et al. (2018b) such as more structured decoding, data augmentation, and using pre-trained representations (they use GloVe (Pennington et al., 2014)) for tokens, columns, and tables.

Our results were surprisingly worse when attempting to incorporate BERT. Of course, successfully incorporating pre-trained representations is not always straightforward. In general, we found using BERT within our architecture to be sensitive to learning rates and learning rate schedules. Notably, the evaluation setup for SPIDER is very different than training, as examples are for tables unseen during training. Models may not generalize well to unseen tables and columns. It's likely that successfully incorporating BERT for SPIDER would require careful tuning of hyperparameters specifically for the database split configuration.

Entity Spans and Relations Ablating span relations between entities and tokens for GEO and ATIS is shown in Table 4. The impact is more significant for ATIS, which contains many queries with multiple entities of the same type, such as *nonstop flights seattle to boston* where disambiguating the origin and destination entities requires knowledge of which tokens they are associated with, given that we represent entities based only on their types for these tasks. We leave for future work consideration of edges between entity candidates that incorporate relevant domain knowledge for these tasks.

Edge Ablations	GEO	ATIS
GNN w/ edge vectors	89.3	87.1
– entity span edges	88.6	34.2

Table 4: Results for ablating information about entity candidate spans for GEO and ATIS.

For SPIDER, results ablating relations between entities and tokens, and relations between entities, are shown in Table 5. This demonstrates the importance of entity relations, as they include useful information for disambiguating entities such as which columns belong to which tables, and which columns have foreign key relations.

Edge Ablations	SPIDER
GNN w/ edge vectors	32.1
– entity span edges	27.8
– entity relation edges	26.3

Table 5: Results for ablating information about relations between entity candidates and tokens for SPIDER.

Edge Representations Using additive edge vectors outperforms using learned edge matrix transformations for implementing f, across all tasks. While the vector formulation is less expressive, it also introduces far fewer parameters per edge type, which can be an important consideration given that our graph contains many similar edge labels, such as those representing similar relative positions between tokens. We leave further exploration of more expressive edge representations to future work. Another direction to explore is a

heterogeneous formulation of the GNN sub-layer, that employs different formulations for different subsets of nodes, e.g. for tokens and entities.

6 Conclusions

We have presented an architecture for semantic parsing that uses a Graph Neural Network (GNN) to condition on a graph of tokens, entities, and their relations. Experimental results have demonstrated that this approach can achieve competitive results across a diverse set of tasks, while also providing a conceptually simple way to incorporate entities and their relations during parsing.

For future direction, we are interested in exploring constrained decoding, better incorporating pre-trained language representations within our architecture, conditioning on additional relations between entities, and different GNN formulations.

More broadly, we have presented a flexible approach for conditioning on available knowledge in the form of entities and their relations, and demonstrated its effectiveness for semantic parsing.

Acknowledgments

We would like to thank Karl Pichotta, Zuyao Li, Tom Kwiatkowski, and Dipanjan Das for helpful discussions. Thanks also to Ming-Wei Chang and Kristina Toutanova for their comments, and to all who provided feedback in draft reading sessions. Finally, we are grateful to the anonymous reviewers for their useful feedback.

References

D. Alvarez-Melis and T. Jaakkola. 2017. Tree structured decoding with doubly recurrent neural networks. In *International Conference on Learning Representations (ICLR)*.

Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic parsing as machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 47–52.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Simaan. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 273–283.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544.

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 *Proceedings of the workshop on Human Language Technology*, pages 43–48. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *ACL*.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272.

Omer Goldman, Veronica Latcinnik, Udi Naveh, Amir Globerson, and Jonathan Berant. 2017. Weakly-supervised semantic parsing with abstract examples. *arXiv preprint arXiv:1711.05240*.

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

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

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

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)*, volume 1, pages 12–22.

Rohit J Kate, Yuk Wah Wong, and Raymond J Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, page 1062. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1545–1556.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in ccg grammar induction for semantic parsing. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1512–1523. Association for Computational Linguistics.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2016. Gated graph sequence neural networks. In *International Conference on Learning Representations (ICLR)*.

Percy Liang, Michael I Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.

Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, Fumin Wang, and Andrew Senior. 2016. Latent predictor networks for code generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 599–609.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515.

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.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1139–1149.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 464–468.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4231–4242.

Lappoon R Tang and Raymond J Mooney. 2000. Automated construction of database interfaces: Integrating statistical and relational learning for semantic parsing. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13*, pages 133–141. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations (ICLR)*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015a. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015b. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2773–2781.

Adrienne Wang, Tom Kwiatkowski, and Luke Zettlemoyer. 2014. Morpho-syntactic lexical generalization for ccg semantic parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1284–1295.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1341–1350.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. In *International Conference on Learning Representations (ICLR)*.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

Yi Yang and Ming-Wei Chang. 2015. S-mart: Novel tree-based structured learning algorithms applied to tweet entity linking. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 504–513.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 201–206.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 440–450.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018a. Typesql: Knowledge-based type-aware neural text-to-sql generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 588–594.

Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. 2018b. Syntaxsqlnet: Syntax tree networks for complex and cross-domaintext-to-sql task. *arXiv preprint arXiv:1810.05237*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018c. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the thirteenth national conference on Artificial intelligence-Volume 2*, pages 1050–1055. AAAI Press.

Luke S Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 658–666. AUAI Press.

Luke S Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. *EMNLP-CoNLL 2007*, page 678.

Kai Zhao and Liang Huang. 2015. Type-driven incremental semantic parsing with polymorphism. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1416–1421.

A Supplemental Material

A.1 Entity Candidate Generator Details

In this section we provide details of how we constructed entity candidate generators for each task.

GEO The annotator was constructed from the geobase database, which provides a list of geographical facts. For each entry in the database, we extracted the name as the entity alias and the type (e.g., "state", "city") as its attribute. Since not all cities used in the GEO query set are listed as explicit entries, we also used cities in the state entries. Finally, geobase has no entries around countries, so we added relevant aliases for "USA" with a special "country" attribute. There was 1 example where an entity in the logical form did not appear in the input, leading to the example being dropped from the training set.

In lieu of task-specific edge relations, we used binary edge labels between entities that captured which annotations span the same tokens. However, experiments demonstrated that these edges did not significantly affect performance. We leave consideration of other types of entity relations for these tasks to future work.

ATIS We constructed a lexicon mapping natural language entity aliases in the dataset (e.g., "newark international", "5pm") to unique entity identifiers (e.g. "ewr:ap", "1700:ti"). For ATIS, this lexicon required some manual construction. Each entity identifier has a two-letter suffix (e.g., "ap", "ti") that maps it to a single attribute (e.g., "airport", "time"). We allowed overlapping entity mentions when the entities referred to different entity identifiers. For instance, in the query span "atlanta airport", we include both the city of Atlanta and the Atlanta airport.

Notably there were 9 examples where one of the entities used as an argument in the logical form did not have a corresponding mention in the input utterance. From manual inspection, many of the dropped examples appear to have incorrectly annotated logical forms. These examples were dropped from training set or marked as incorrect if they appeared in the test set.

We use the same binary edge labels between entities as for GEO.

SPIDER For SPIDER we generalize our notion of entities to consider tables and columns as entities. We attempt to determine spans for each table and column by computing normalized Levenshtein distance between table and column names and unigrams or bigrams in the utterance. The best alignment having a score > 0.75 is selected, and we use these generated alignments to populate the edges between tokens and entity candidates.

We generate a set of attributes for the table based on unigrams in the table name, and an attribute to identify the entity as a table. Likewise, for columns, we generate a set of attributes based on unigrams in the column name as well as an attribute to identify the value type of the column. We also include attributes indicating whether an alignment was found between the entity and the input text.

We include 3 task-specific edge label types between entity candidates to denote bi-directional relations between column entities and the table entity they belong to, and to denote the presence of a foreign key relationship between columns.