Compositional Generalization via Semantic Tagging

Hao Zheng and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
`Hao.Zheng@ed.ac.uk` `mlap@inf.ed.ac.uk`

Abstract

Although neural sequence-to-sequence models have been successfully applied to semantic parsing, they fail at *compositional generalization*, i.e., they are unable to systematically generalize to unseen compositions of seen components. Motivated by traditional semantic parsing where compositionality is explicitly accounted for by symbolic grammars, we propose a new decoding framework that preserves the expressivity and generality of sequence-to-sequence models while featuring lexicon-style alignments and disentangled information processing. Specifically, we decompose decoding into two phases where an input utterance is first tagged with semantic symbols representing the meaning of individual words, and then a sequence-to-sequence model is used to predict the final meaning representation conditioning on the utterance *and* the predicted tag sequence. Experimental results on three semantic parsing datasets show that the proposed approach consistently improves compositional generalization across model architectures, domains, and semantic formalisms.[1]

1 Introduction

Semantic parsing aims at mapping natural language utterances to machine-interpretable meaning representations such as executable queries or logical forms. Sequence-to-sequence neural networks (Sutskever et al., 2014) 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, *inter alia*). Despite recent success, there has been mounting evidence (Finegan-Dollak et al., 2018; Keysers et al., 2020; Herzig and Berant, 2021; Lake and Baroni, 2018) that these models fail at *compositional generalization*, i.e, they are unable to systematically generalize to *unseen* compositions of *seen* components. For example, a model that observed at training time the questions "*How many people live in California?*" and "*How many people live in the capital of Georgia?*" fails to generalize to questions such as "*How many people live in the capital of California?*". This is in stark contrast with human language learners who are able to systematically generalize to such compositions (Fodor and Pylyshyn, 1988; Lake et al., 2019).

Training Set
What is the density of Texas?
`select density from state where state_name = "texas"`
Test Set (Question split)
What is the population density of Maine?
`select density from state where state_name = "maine"`
Test Set (Query Split)
How many people live in Washington?
`select population from state where state_name = "washington"`

Table 1: Two test examples from the question- and query-based splits of GeoQuery and a training example included in both splits. The example in the question-based split shares the same query pattern as the training example while the example in the query-based split has a query pattern different from the training example.

Previous work (Finegan-Dollak et al., 2018) has exposed the inability of semantic parsers to generalize compositionally simply by evaluating their performance on different dataset splits. Existing datasets commonly adopt *question-based* splits where many examples in the test set have the same query templates (induced by anonymizing named entities) as examples in the training. As a result, many of the queries in the test set are seen in training, and parsers are being evaluated for their ability to generalize to questions with different surface forms but the same meaning. In contrast, when

[1] Our code and data can be found at `https://github.com/mswellhao/Semantic-Tagging`.

Findings of the Association for Computational Linguistics: EMNLP 2021, pages 1022–1032
November 7–11, 2021. ©2021 Association for Computational Linguistics

adopting a *query-based* split, the structure of the queries in the test set is unobserved at training time, and parsers therefore must generalize to questions with different meanings. Table 1 illustrates the difference between question- and query-based splits on GeoQuery (Zelle and Mooney, 1996).

On the contrary, compositional generalization poses no problem for traditional semantic parsers (Zettlemoyer and Collins, 2005, 2007; Wong and Mooney, 2006, 2007; Liang et al., 2013) which typically use a (probabilistic) grammar; the latter defines the meaning of individual words and phrases and how to best combine them in order to obtain meaning representations for entire utterances. Neural semantic parsers do away with representing symbolic structure explicitly in favor of a more general approach which directly transduces the utterance into a logical form, avoiding domain-specific assumptions and grammar learning.

Nonetheless, the symbolic paradigm provides two important insights that could serve as a guide in designing neural semantic parsers with better compositional generalization. Firstly, the probability of a logical form is decomposed into *local* factors under strong conditional independence assumptions while in neural semantic parsing the prediction of each symbol directly depends on *all* previously decoded symbols. This strong expressivity may hurt compositional generalization since different kinds of information are bundled together, rendering the model's predictions susceptible to irrelevant context changes. Secondly, there exist *hard* alignments between logical constructs and linguistic expressions but in neural parsers the two are only loosely related via the *soft* attention mechanism. Explicit alignments can help distinguish which language segments are helpful for predicting certain components in the logical form, potentially improving compositional generalization.

In this paper, we devise a new decoding framework that preserves the expressivity and generality of sequence-to-sequence models while featuring lexicon-style alignments and disentangled information processing. Specifically, we decompose decoding into two phases. Given a natural language utterance, each word is first labeled with a semantic symbol representing its meaning via a tagger. Semantic symbols are atomic units like predicates (in λ-calculus) or columns (in SQL). The tagger *explicitly* aligns semantic symbols to tokens or token spans in the utterance. Moreover, the prediction of each semantic symbol is conditionally independent of other symbols in the logical form. This is reminiscent of lexicons in classical semantic parsers, but a major difference is that our tagger is a neural model which considers information based on the entire utterance and can generalize to new words. A sequence-to-sequence model takes the utterance and predicted tag sequence which serves as a soft constraint on the output space, and generates the final meaning representation. Our framework is general in that it could incorporate any sequence-to-sequence model as the base model and augment it with semantic tagging.

We evaluate the proposed approach on query-based splits of three semantic parsing benchmarks: Atis, GeoQuery, and a subset of WikiSQL covering different semantic formalisms (λ-calculus and SQL). We report experiments with LSTM- and Transformer-based models (Dong and Lapata, 2016, 2018; Vaswani et al., 2017) demonstrating that our framework improves compositional generation across datasets and model architectures. Our approach is also superior to a recent data augmentation proposal (Andreas, 2020), specifically designed to enhance compositional generalization.

2 Related Work

The realization that neural sequence models perform poorly in settings requiring compositional generalization has led to several research efforts aiming to study the extent of this problem and how to handle it. For instance, recent studies have proposed benchmarks which allow to measure different aspects of compositional generalization.

Lake and Baroni (2018) introduce SCAN, a grounded navigation task where a learner must translate natural language commands into a sequence of actions in a synthetic language. Bahdanau et al. (2019) use a synthetic VQA task to evaluate whether models can reason about all possible object pairs after training only on a small subset. They show that modular structured models are best in terms of systematic generalization, while end-to-end versions do not generalize as well. Keysers et al. (2020) introduce a method to systematically construct benchmarks for evaluating compositional generalization. Using Freebase as an example, they create questions which maximize compound divergence (e.g., combinations of entities and relations) while guaranteeing that the atoms (aka the primitive elements used to compose these questions) remain the same between train and test sets.

Other work proposes data augmentation as a way of injecting a compositional inductive bias into neural sequence models. Under this protocol, synthetic examples are constructed by taking real training examples and replacing (possibly discontinuous) fragments with other fragments that appear in at least one similar environment. Recombination operations can be performed by applying rules (Andreas, 2020) or learned using a generative model (Aky). Herzig and Berant (2021) follow a more traditional approach (Zelle and Mooney, 1996; Ge and Mooney, 2005; Zettlemoyer and Collins, 2005; Wong and Mooney, 2006, 2007; Zettlemoyer and Collins, 2007; Kwiatkowksi et al., 2010; Kwiatkowski et al., 2011) and develop a span-based parser which predicts a tree over an input utterance, explicitly encoding how partial programs compose over spans in the input. Finally, Oren et al. (2020) improve compositional generalization with the use of contextual representations, extensions to decoder attention, and downsampling examples from frequent templates.

We decompose decoding in two stages where the input is first tagged with semantic symbols which are then subsequently used to predict the final meaning representation. These semantic tags are automatically induced from logical forms without any extra annotation and vary depending on the meaning representation at hand (e.g., λ-calculus, SQL). They serve the goal of injecting inductive bias for compositional generalization rather than expressing general semantic information across languages (see Abzianidze and Bos 2017 for a proposal to develop a universal semantic tagset for non-executable semantic parsing). Our framework can be applied to different sequence-to-sequence models, domains, and semantic formalisms. It does not require manual task-specific engineering (Herzig and Berant, 2021) and is orthogonal to data augmentation methods (Andreas, 2020; Aky) and other extensions (Oren et al., 2020) which we could also incorporate.

3 Model Architecture

Our goal is to learn a semantic parser that takes as input a natural language utterance $x = x_1, x_2, ..., x_n$ and predicts a meaning representation $y = y_1, y_2, ..., y_m$. We decompose the parser $p(y|x)$ into a two-stage generation process:

$$p(y|x) = p(y|x,z)p(z|x) \tag{1}$$

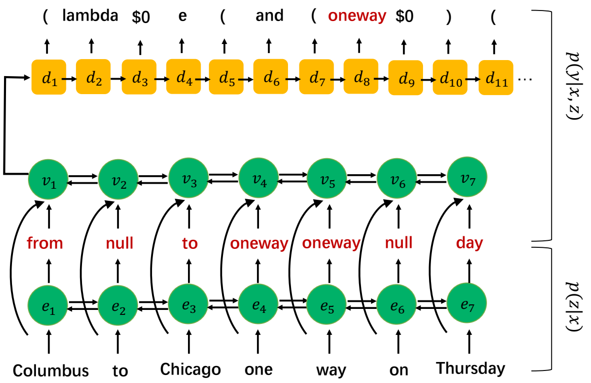
(lambda $0 e (and (oneway $0) (
d_1 d_2 d_3 d_4 d_5 d_6 d_7 d_8 d_9 d_{10} d_{11} ...
v_1 v_2 v_3 v_4 v_5 v_6 v_7
from null to oneway oneway null day
e_1 e_2 e_3 e_4 e_5 e_6 e_7
Columbus to Chicago one way on Thursday
$p(y|x,z)$
$p(z|x)$

Figure 1: We first tag natural language input x with semantic symbols (e.g., predicates) and predict tag sequence z. We generate the final semantic representation y, given x and z as input.

where $z = z_1, z_2, ..., z_n$ is a tag sequence for x. Every tag z_t is a symbol in y representing the meaning of x_t. Therefore, the first-stage model $p(z|x)$ is essentially a tagger that tries to predict the semantics of individual words. The second-stage model takes word sequence x and its accompanying tag sequence z as input, and generates the final semantic representation y. Figure 1 shows the two-stage generation process. It is important to note that tags z are *latent* and must be induced from training data, i.e., pairs of natural language utterances and representations of their meaning. We discuss how the tagger is learned in Section 4.

3.1 Semantic Tagging

As shown in Figure 1, the tagging model $p(z|x;\theta)$ contains an encoder which transforms input sequence $x_1, x_2, ..., x_n$ into a sequence of context-sensitive vector representations $\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n$. Each word x_i is mapped to embedding $\mathbf{w}_i$, and the sequence of word embeddings $\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_n$ is fed to a bi-directional recurrent neural network with long short-term memory (LSTM) units (Hochreiter and Schmidhuber, 1997). A bi-LSTM recursively computes the hidden states at the t-th time step via:

$$\overrightarrow{\mathbf{h}}_i = f_{\text{LSTM}} \quad (\overrightarrow{\mathbf{h}}_{i-1}, \mathbf{w}_i) \tag{2}$$

$$\overleftarrow{\mathbf{h}}_i = f_{\text{LSTM}} \quad (\overleftarrow{\mathbf{h}}_{i+1}, \mathbf{w}_i) \tag{3}$$

$$\mathbf{h}_i = [\overrightarrow{\mathbf{h}}_i, \overleftarrow{\mathbf{h}}_i] \tag{4}$$

where $\mathbf{h}_i$ is the concatenation of vectors $\overrightarrow{\mathbf{h}}_i$ and $\overleftarrow{\mathbf{h}}_i$, and f_{LSTM} refers to the LSTM function. We feed both $\mathbf{h}_i$ and $\mathbf{w}_i$ to the final output layer in order to predict tags z:

$$
\begin{aligned}
p(z|x;\theta) &= \prod_{i=1}^{n} p(z_i|x;\theta) && (5)\\
&= \prod_{i=1}^{n} \text{softmax}(\mathbf{W}\mathbf{h}_i + \mathbf{U}\mathbf{w}_i + \mathbf{b}) && (6)
\end{aligned}
$$

$\mathbf{W}$, $\mathbf{U}$, and $\mathbf{b}$ are parameters in the output layer.

3.2 Meaning Representation Generation

LSTM-based encoder-decoder models with an attention mechanism have been successfully applied to a wide range of semantic parsing benchmarks (Dong and Lapata, 2016; Jia and Liang, 2016; Iyer et al., 2017), while Transformers have been rapidly gaining popularity for various NLP tasks including semantic parsing (Wang et al., 2020; Sherborne et al., 2020). Our approach is model-agnostic in that it could be combined with any type of sequence-to-sequence model; to highlight this versatility, we present experiments with both LSTM- and Transformer-based models. We first embed the predicted tag and word sequences, obtaining tag embeddings $\mathbf{e}_1^g, \mathbf{e}_2^g, .., \mathbf{e}_n^g$ and word embeddings $\mathbf{e}_1^w, \mathbf{e}_2^w, ..., \mathbf{e}_n^w$. Then, we concatenate the two types of embeddings at each time step and feed them to a sequence-to-sequence model:

$$
\begin{aligned}
\mathbf{u}_t &= [\mathbf{e}_t^g, \mathbf{e}_t^w] && (7)\\
y &= f_{\texttt{seq2seq}}(\mathbf{u}) && (8)
\end{aligned}
$$

where $[\cdot, \cdot]$ denotes vector concatenation and $f_{\texttt{seq2seq}}$ denotes a sequence-to-sequence model variant (LSTM- or Transformer-based in our case) that takes a sequence of vector representations as input and ultimately generates a logical form. Tag embeddings are shared with the embeddings used in the decoder. Therefore, the only adaptation we make to the baseline model is replace the original word embeddings with tag-augmented input.

4 Model Training

Our proposed approach combines a semantic tagger with a sequence-to-sequence model. The tagger learning problem is challenging since z is unobserved. In this section, we explain how the tagger and the overall model are trained.

4.1 Tagger Learning

We learn a tagger $p(z|x;\theta)$ from training data consisting of pairs of natural language utterances $x = x_1, x_2, ..., x_n$ and symbol sets $s = \{s_1, s_2, ..., s_l\}$ (with $s_j \in y$). The symbol set contains atomic semantic units such as λ-calculus predicates and SQL column names. Table 2 presents examples of symbol sets for these two formalisms. As can be seen, symbols have close ties to utterances, there is often a correspondence between them and individual words or phrases. It is therefore natural to predict this (basic) part of a meaning representation via a tagger. To bridge the gap between the tag sequence we intend to predict and the symbol set we have as supervision, we introduce latent variable $a = a_1, a_2, ..., a_n$ where a_j denotes the index of a word aligned to s_j. We add $(n - l)$ `null` symbols to target set $s = \{s_1, s_2, ..., s_l, s_{l+1}, ..., s_n\}$ because n is typically larger than l, and we allow the tagger to output `null` for some words.

	λ-calculus
x :	Columbus to Chicago one way on Thursday
z :	Columbus/**`from`** to/**`null`** Chicago/**`to`** one/**`oneway`** way/**`oneway`** on/**`null`** Thursday/**`day`**
s :	**`oneway, from, to, day`**
y :	`( lambda $0 e ( and (` **`oneway`** `$0 ) (` **`from`** `$0 columbus:ci ) (` **`to`** `$0 chicago:ci ) (` **`day`** `$0 thursday:da ) ) )`
	SQL
x :	What is the area of Washington
z :	What/**`null`** is/**`null`** the/**`null`** area/**`area`** of/**`null`** Washington/**`state_name`**
s :	**`area, state_name`**
y :	`select` **`area`** `from state where` **`state_name`** `= "washington"`

Table 2: Utterances x, their meaning representations y, symbol sets s, and predicted word/**`tag`** sequences z.

Entity Linking For some symbols, it is rather straightforward to determine the corresponding alignments based on the results of entity linking, a critical subtask in semantic parsing which is generally treated as a preprocessing step (Dong and Lapata, 2016; Jia and Liang, 2016). We thus define the following two rules to automatically align symbols to words in an utterance based on entity linking: (1) for λ-calculus expressions, if a predicate takes only one entity as an argument (e.g., `day $0 thursday:da`) and this entity can be linked to a word or phrase in the utterance, we assume there is an alignment between them (e.g., `day` aligns to *Thursday*); (2) for SQL expressions, if the entity in a filter clause (e.g., `state_name = "washington"`) can be linked to an expression in the utterance, again we align the column (e.g., `state_name`) to the linguistic expression (e.g., *Washington*). Both rules capture the intuition that some semantic symbols are implied by corresponding entities without being explicitly verbalized. As shown in Table 2, there is no linguistic expression in the utterance "*What is the area of Washington*" which corresponds to the logical expression of `state_name`, instead `state_name` is implied by the entity `washington`.

Expectation-Maximization Besides entity linking, there remain symbols without alignments, such as unary predicates (e.g., `oneway $0`). For these, we use an EM-style algorithm which iteratively infers latent alignments a and uses them to update the tagger. A hard-EM algorithm that predicts the most probable a seems reasonable as in most cases there is a single correct alignment. However, we find that hard-EM renders training unstable and prone to overfitting to incorrect alignments. We instead warm up the training with a soft-EM algorithm first and switch to hard-EM later on. Without loss of generality, we describe the algorithm for all symbols including those that could be aligned via (entity linking) rules. Specifically, we model the generation of s as follows:

$$\begin{aligned} p(s|x;\theta) &= \sum_a p(a|x)p(s|x,a;\theta) \\ &= \sum_a p(a|x) \prod_{j=1}^{n} p(z_{a_j} = s_j|x;\theta) \quad (9) \end{aligned}$$

where $p(a|x)$ is a uniform prior over a and $p(z_{a_j} = s_j|x;\theta)$ is the tagger model above. We could constrain the alignment from words to symbols to be injective (as this would more faithfully capture the complex dependencies between them). Unfortunately, this renders posterior inference on a intractable. Instead, we model the alignment of each symbol independently as:

$$\begin{aligned} p(s|x;\theta) &= \sum_a \prod_{j=1}^{n} p(a_j|x) \prod_{j=1}^{n} p(z_{a_j} = s_j|x;\theta) \\ &= \prod_{j=1}^{n} \sum_{a_j} p(a_j|x)p(z_{a_j} = s_j|x;\theta) \quad (10) \end{aligned}$$

Under this assumption, we are able to exactly compute the posterior probability of each alignment a_j:

$$\begin{aligned} \pi_{ij}(\theta) &= p(a_j = i|x,s;\theta) \\ &= \frac{p(a_j = i|x)p(z_i = s_j|x;\theta)}{\sum_{\tilde{i}=1}^{n} p(a_j = \tilde{i}|x)p(z_{\tilde{i}} = s_j|x;\theta)} \quad (11) \end{aligned}$$

Note that we manually set the value of $\pi_{ij}(\theta)$ if a_j can be induced in advance via entity linking. At the t-th iteration, we first use the present tagger $p(z|x;\theta^t)$ to compute $\pi_{ij}(\theta^t)$, the likelihood of aligning symbol s_j to word x_i. For soft-EM, these assignments are then directly used to train the tagger with the following objective:

$$\begin{aligned} \mathcal{J}_t(\theta) &= \sum_{i=1}^{n}\sum_{j=1}^{n} \pi_{ij}(\theta^t) \log p(z_i = s_j|x;\theta) \quad (12) \\ \theta^{t+1} &= \arg\max_\theta \ \mathcal{J}_t(\theta) \quad (13) \end{aligned}$$

For hard-EM, one could exploit $\pi_{ij}(\theta^t)$ to induce the most probable alignment for each symbol. However, there are cases where a symbol is aligned to multiple words, e.g., when the same word occurs multiple times in an utterance or when a symbol is aligned to a phrase. To deal with such cases, we induce a hard-version of the posterior probability $\tilde{\pi}_{ij}(\theta^t)$ in the following way:

$$\begin{aligned} \tilde{\pi}_{ij}(\theta^t) &= \begin{cases} 1 & \text{if } \pi_{ij}(\theta^t) > \beta \\ 0 & \text{otherwise} \end{cases} \quad (14) \\ & \qquad (1 \le j \le l) \\ \tilde{\pi}_{ij}(\theta^t) &= \frac{1 - \sum_{k=1}^{l} \tilde{\pi}_{ik}}{n-l} \quad (15) \\ & \qquad (l+1 \le j \le n) \end{aligned}$$

where β is a threshold used to discretize the soft alignment distributions. Reshaping the posteriors in this manner allows a symbol to be aligned to multiple words while removing noisy incorrect alignments. Equation (15) ensures that the sum of posteriors corresponding to a word is one, in the hope of encouraging the predicted tag sequence distribution to be as close to a normal tag sequence distribution as possible. We replace $\pi_{ij}(\theta^t)$ in $\mathcal{J}(\theta|\theta^t)$ with $\tilde{\pi}_{ij}(\theta^t)$ as the training objective to perform hard-EM updates:

$$\begin{aligned} \tilde{\mathcal{J}}_t(\theta) &= \sum_{i=1}^{n}\sum_{j=1}^{n} \tilde{\pi}_{ij}(\theta^t) \log p(z_i = s_j|x;\theta) \quad (16) \\ \theta^{t+1} &= \arg\max_\theta \ \tilde{\mathcal{J}}_t(\theta) \quad (17) \end{aligned}$$

Our training procedure is shown in Algorithm 1. Note that in each EM iteration, we use objective $\mathcal{J}_t(\theta)$ or $\tilde{\mathcal{J}}_t(\theta)$ to compute the gradient and update parameters once rather than maximizing the objective function.

Algorithm 1: Training the tagger

```
Input: Dataset $\mathcal{D}$ where each example is a
       question $x$ paired with symbol set $s$.
       Number of soft-EM updates $T_s$.
       Number of overall updates $T$.
Output: Tagger model parameters $\theta^{T+1}$
 Initialize tagger parameters $\theta^1$ randomly;
for $t = 1, ..., T$ do
    sample an example ($x$, $s$)
    if $t < T_s$ then
        /* do soft-EM update  */
        Compute $\pi_{ij}(\theta^t)$
        $\theta^{t+1} \leftarrow$ Optimizer($\theta^t, \nabla_{\theta_t}\mathcal{J}_t(\theta_t)$)
    else
        /* do hard-EM update  */
        Compute $\tilde{\pi}_{ij}(\theta^t)$
        $\theta^{t+1} \leftarrow$ Optimizer($\theta^t, \nabla_{\theta_t}\tilde{\mathcal{J}}_t(\theta_t)$)
    end
end
return $\theta^{T+1}$
```

4.2 Parser Learning

Learning a semantic parser in our setting is straightforward. After training the tagger, we run it over the examples in the training data and obtain tag sequence $\hat{z}$ for each pair of utterance x and meaning representation y.

$$\hat{z} = \arg\max_z p(z|x;\theta) \tag{18}$$

Then, we maximize the likelihood of generating y given x and $\hat{z}$:

$$\hat{\theta} = \arg\max_\theta \log p(y|x,\hat{z};\theta) \tag{19}$$

5 Experimental Setup

Datasets Our experiments evaluate the proposed framework on compositional generalization. We present results on query-based splits for three widely used semantic parsing benchmarks, namely ATIS (Dahl et al., 1994), GEOQUERY (Zelle and Mooney, 1996), and WIKISQL (Zhong et al., 2017). For GEOQUERY (880 language queries to a database of U.S. geography) and ATIS (5,410 queries to a flight booking system) meaning representations are in λ-calculus and SQL. We adopt the split released by Finegan-Dollak et al. (2018) for SQL. We create query-based splits for λ-calculus, as we use the preprocessed versions provided in Dong and Lapata (2018), where natural language expressions are lowercased and stemmed with NLTK (Bird et al., 2009), and entity mentions are replaced by numbered markers.

WIKISQL is a large-scale semantic parsing dataset released more recently (Zhong et al., 2017). It is used as a testbed for generating an SQL query given a natural language question and table schema (i.e., table column names) without using the content values of tables. Since SQL queries in most examples are simple and only contain one filtering condition, we use a subset (16,835 training examples, 2,602 validation examples, and 4,915 test examples) containing queries with more than one filtering condition. These examples are more compositional and better suited to evaluating compositional generalization.

Comparison Models On ATIS and GEOQUERY we trained two baseline sequence-to-sequence models which we implemented using LSTMs and Transformers as the base units (see Section 3.2). To examine whether our results carry over to pretrained contextual representations, we report experiments with an LSTM model enhanced with RoBERTa (Liu et al., 2019). We also compare against two related approaches. The first is GECA (Andreas, 2020), a recently proposed data augmentation method aimed at providing a compositional inductive bias into sequence-to-sequence models. The second is Attention Supervision introduced in Oren et al. (2020). They encourage generalization by supervising the decoder attention with precomputed token alignments. We use the alignments induced by our tagger instead of an off-the-shelf word aligner adopted in their paper.

For WIKISQL, our baseline model follows the COARSE2FINE approach put forward in Dong and Lapata (2018) which is well suited to the formulaic nature of the queries, takes the table schema into account, and performs on par with some more sophisticated models (McCann et al., 2018; Yu et al., 2018). They predict `select` and `where` SQL clauses separately (all queries in WIKISQL follow the same format, i.e., "`SELECT agg_op agg_col where (cond_col cond_op cond AND)...`", which is a small subset of the SQL syntax). The `select` clause is predicted via two independent classifiers, while the `where` clause is generated via a sequence model with a sketch as an intermediate outcome. Their encoder augments question representations with table information by computing attention over table column vectors and deriving a context vector to summarize the relevant columns for each word.

Our tagger uses Coarse2Fine's table-aware encoder to predict tags. Our parser diverges slightly from their model: while for each word the context vector is originally computed by the attention mechanism, we replace it with the column vector specified by the corresponding tag.

Configuration We implemented the base semantic parsers (LSTM and Transformer) with fairseq (Ott et al., 2019). As far as GECA is concerned, we have a different setting from Andreas (2020): we use the preprocessed versions provided by Dong and Lapata (2018) for Atis and GeoQuery, while they report experiments on GeoQuery only, with different preprocessing. We used their open-sourced code to generate synthetic data for our setting in order to make experiments comparable. For Coarse2Fine (Dong and Lapata, 2018), we used the code released by the authors.

Hyperparameters for the semantic taggers were validated on the development split of Atis and were directly copied for GeoQuery because of its small size. Dimensions of hidden vectors and word embeddings were selected from {150, 200, 250, 300}. The number of layers was selected from {1, 2}. Batch size was set to 20 and the overall update step was set to 20,000. The number of steps for soft-EM updates was selected from {5,000, 7,000, 10,000, 13,000}. The threshold β used in hard-EM was selected from {0.20, 0.23, 0.26, 0.29, 0.32, 0.35}. We used the Adam optimizer (Kingma and Ba, 2015) to train the models and the learning rate was selected from {0.0001, 0.0003, 0.001}. Our semantic parsers used the same hyperparameters as the base models except for some necessary changes to incorporate tag inputs. For models using RoBERTa, we first freeze RoBERTa and train the model for some steps, and then resume fine-tuning.

Evaluation We use exact-match accuracy as our evaluation metric, namely the percentage of examples that are correctly parsed to their gold standard meaning representations. For WikiSQL, we also execute generated SQL queries on their corresponding tables, and report execution accuracy which is defined as the proportion of correct answers.

Method	**λ-calculus**		**SQL**	
	Geo	Atis	Geo	Atis
GECA	48.1	51.6	52.1	24.0
Transformer	39.8	51.2	53.9	23.0
Transformer + AS	43.4	53.3	58.6	22.0
Transformer + ST	44.0	53.0	61.9	28.6
LSTM	49.8	56.2	48.5	28.0
LSTM + AS	53.6	59.7	46.9	28.7
LSTM + ST	52.1	62.1	63.6	29.1
RoBERTa	54.4	57.5	58.8	28.6
RoBERTa + AS	56.3	59.9	59.3	28.4
RoBERTa + ST	57.5	63.7	69.6	27.7

Table 3: Exact-match accuracy on GeoQuery and Atis; results averaged over 5 random seeds; ST stands for semantic tagging; AS is attention supervision.

Method	**Acc**	**Exe**	**`where`**
Coarse2Fine	58.0	68.2	71.3
Coarse2Fine + AS	58.8	69.2	72.8
Coarse2Fine + ST	60.6	71.3	75.0

Table 4: Evaluation results on a WikiSQL subset. **Acc**: exact-match accuracy; **Exe**: execution accuracy; **`where`**: accuracy of predicting `where` clauses.

6 Results

Does Tagging Help Parsing? Table 3 summarizes our results on Atis and GeoQuery. On both datasets, we observe that the proposed tagger (+ ST) boosts the performance of the base model (Transformer, LSTM) for both λ-calculus and SQL. The LSTM is generally superior to Transformer except on SQL GeoQuery. Enhancing the LSTM with pretrained contextual representations (see the last block in the table) generally increases accuracy, yet our semantic tagger brings improvements on top of RoBERTa (with the exception of SQL Atis). This points to the generality of our approach which benefits neural parsers with different architectures trained on distinct semantic representations. Gains are particularly significant on Atis with λ-calculus (we observe an absolute improvement of 6.2 points over RoBERTa) and GeoQuery with SQL (with 10.8 points absolute improvement over RoBERTa).

In some settings, attention supervision (+AS) also achieves improvements over baseline sequence models, but these are inconsistent and sometimes it even slightly hurts performance. We find that attention supervision is sensitive to the weight hyperpa-

Method	λ-calculus		SQL	
	Geo	Atis	Geo	Atis
LSTM	16.1 / 10.9 / 23.2	13.7 / 9.8 / 20.1	14.1 / 5.6 / 31.8	21.3 / 26.5 / 23.7
LSTM + ST	16.5 / 9.7 / 21.7	13.0 / 8.9 / 15.9	19.0 / 6.7 / 10.5	22.1 / 24.9 / 23.9
RoBERTa + ST	13.0 / 9.6 / 19.7	12.9 / 6.9 / 16.4	12.7 / 5.7 / 11.8	22.6 / 16.6 / 32.8

Table 5: Breakdown of different types of error. In each cell, left shows the proportion of predicting correct semantic symbols but incorrect queries; middle is the proportion of predicting a subset of correct symbols (i.e., missing some semantic symbols); right is the proportion of predicting symbols which do not exist in gold queries.

rameter that controls the strength of attention loss and requires careful tuning to achieve good performance. We conjecture that the soft attention mechanism (even with proper supervision signals) is still sensitive to irrelevant context changes and prone to errors in cases requiring compositional generalization. The LSTM+ST model achieves better accuracy than GECA which adopts a data augmentation strategy to train a LSTM-based sequence-to-sequence model for compositional generalization. We incorporate a similar inductive bias into the parser, but in an orthogonal way.

Results on WikiSQL are shown in Table 4. Semantic tagging boosts coarse2fine in terms of exact match and execution accuracy. In particular, it improves the prediction of `where` clauses, by a 4.3% absolute margin. We would not expect semantic tagging to benefit any other parts of the generation of the SQL query, since only `where` clauses are decoded sequentially in the coarse2fine model. Gains in the generation of `where` clauses translate to improvements in overall accuracy. Attention supervision (+AS) also improves generalization but falls behind our semantic tagger.

Do Meaning Representations Matter? Improvements of our semantic tagger on Atis with SQL and GeoQuery with λ-calculus are less dramatic compared to Atis with λ-calculus and GeoQuery with SQL. Upon closer inspection, we find that Atis SQL queries typically include many bridging columns that are used to join two tables. This arises from the complex database structure in Atis: there are 32 tables in total and each query involves 6.4 tables on average. These bridging columns are SQL-specific and generally do not align with any linguistic expressions, so we cannot improve their prediction via semantic tagging. A prerequisite for semantic tagging is that there exist alignments between language expressions and atomic semantic symbols. We could restrict the semantic tagger to only predicting symbols which align to linguistic expressions and leave the generation of other symbols to the second stage. However, how to automatically select appropriate symbols as semantic tags is an avenue for future work.

On GeoQuery with λ-calculus, the semantic tagger performs extremely well, achieving 86.2% accuracy in predicting semantic symbols, but the final accuracy in predicting queries is only 52.1% (LSTM+ST). Although semantic tagging can help generalize to utterances where seen syntactic structure and concept words are combined in an unseen way (e.g., *Monkeys like bananas* generalizes to *Cats like fish*), it fails to generalize to utterances with unseen syntactic structure (e.g., *Monkeys like bananas* generalizes to *Cats like fish that like water*). Handling utterances with unseen composition of seen syntactic components is yet another generalization challenge for modern semantic parsers.

Where do Gains Come from? Our approach transfers much of the prediction of semantic symbols from the sequence-to-sequence model to the tagger; it does this by replacing the attention mechanism, which learns to attend to *specific* parts of an utterance, with per-word tagging which considers *all* parts of an utterance. We hypothesize that this architecture can better exploit source information to predict individual semantic symbols. To test this hypothesis, we analyzed errors in the predictions of the LSTM model with and without the proposed semantic tagger, and classified them into three types. The first type predicts incorrect queries but correct semantic symbols. The second type predicts only a subset of correct semantic symbols, thus omitting some semantic symbols. The third type predicts wrong semantic symbols that do not exist in gold queries. As shown in Table 5, semantic tagging mainly reduces the errors of predicting wrong semantic symbols, while in some cases it can lead to a modest increase in the first type of errors. Overall, semantic tagging improves the prediction of individual semantic symbols even though this improvement does not always translate into more accurate queries.

7 Conclusions

We presented a two-stage decoding framework, aiming to improve compositional generalization in neural semantic parsing. Central to our approach is a semantic tagger which labels the input with semantic symbols representing the meaning of individual words. A neural sequence-to-sequence parsing model consider the input utterance and the predicted tag sequence to generate the final meaning representation. Our framework can be combined with different neural models and semantic formalisms and demonstrates superior performance to related compositional generalization approaches (Andreas, 2020; Oren et al., 2020). In the future, we would like to extend our approach to learning syntactic generalizations.

Acknowledgments We thank the anonymous reviewers for their feedback. We gratefully acknowledge the support of the European Research Council (award number 681760).

References

Lasha Abzianidze and Johan Bos. 2017. Towards universal semantic tagging. In *IWCS 2017 — 12th International Conference on Computational Semantics — Short papers*.

Jacob Andreas. 2020. Good-enough compositional data augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7556–7566, Online. Association for Computational Linguistics.

Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron Courville. 2019. Systematic generalization: What is required and can it be learned? In *Proceedings of the 7th International Conference on Learning Representations*, New Orleans, Louisiana.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

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, Stroudsburg, PA, USA.

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.

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

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-SQL evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360, Melbourne, Australia. Association for Computational Linguistics.

Jerry A. Fodor and Zenon W. Pylyshyn. 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 1-2(28):3–71.

Ruifang Ge and Raymond Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 9–16, Ann Arbor, Michigan. Association for Computational Linguistics.

Jonathan Herzig and Jonathan Berant. 2021. Span-based semantic parsing for compositional generalization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 908–921, Online. Association for Computational Linguistics.

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

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.

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.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz

Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations, ICLR 2020*, Online.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015*, San Diego, California.

Tom Kwiatkowksi, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1223–1233, Cambridge, MA. Association for Computational Linguistics.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1512–1523, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Brenden Lake, Tal Linzen, and Marco Baroni. 2019. Human few-shot learning of compositional instructions. In *Proceedings of the 41st Annual Meeting of the Cognitive Science Society*, pages 611–617, Montréal, Canada.

Brenden M. Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2879–2888, Stockholm, Sweden. PMLR.

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

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *CoRR*, abs/1806.08730.

Inbar Oren, Jonathan Herzig, Nitish Gupta, Matt Gardner, and Jonathan Berant. 2020. Improving compositional generalization in semantic parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2482–2495, Online. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Tom Sherborne, Xu Yumo, and Mirella Lapata. 2020. Bootstrapping a crosslingual semantic parser. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 499–517, Online. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

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.

Yuk Wah Wong and Raymond Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 439–446, New York City, USA. Association for Computational Linguistics.

Yuk Wah Wong and Raymond Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 960–967, Prague, Czech Republic. Association for Computational Linguistics.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018. 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)*, pages 588–594, New Orleans, Louisiana. Association for Computational Linguistics.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the 13th National Conference on Artificial Intelligence - Volume 2*, page 1050–1055. AAAI Press.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 678–687, Prague, Czech Republic. Association for Computational Linguistics.

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*, page 658–666, Arlington, Virginia, USA. AUAI Press.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.