Data Recombination for Neural Semantic Parsing

Robin Jia
Computer Science Department
Stanford University
robinjia@stanford.edu

Percy Liang
Computer Science Department
Stanford University
pliang@cs.stanford.edu

Abstract

Modeling crisp logical regularities is crucial in semantic parsing, making it difficult for neural models with no task-specific prior knowledge to achieve good results. In this paper, we introduce data recombination, a novel framework for injecting such prior knowledge into a model. From the training data, we induce a high-precision synchronous context-free grammar, which captures important conditional independence properties commonly found in semantic parsing. We then train a sequence-to-sequence recurrent network (RNN) model with a novel attention-based copying mechanism on datapoints sampled from this grammar, thereby teaching the model about these structural properties. Data recombination improves the accuracy of our RNN model on three semantic parsing datasets, leading to new state-of-the-art performance on the standard GeoQuery dataset for models with comparable supervision.

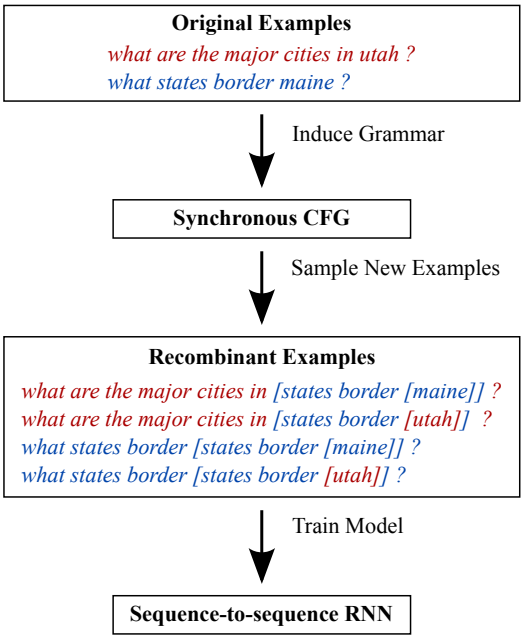

Figure 1: An overview of our system. Given a dataset, we induce a high-precision synchronous context-free grammar. We then sample from this grammar to generate new "recombinant" examples, which we use to train a sequence-to-sequence RNN.

1 Introduction

Semantic parsing—the precise translation of natural language utterances into logical forms—has many applications, including question answering (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Zettlemoyer and Collins, 2007; Liang et al., 2011; Berant et al., 2013), instruction following (Artzi and Zettlemoyer, 2013b), and regular expression generation (Kushman and Barzilay, 2013). Modern semantic parsers (Artzi and Zettlemoyer, 2013a; Berant et al., 2013) are complex pieces of software, requiring hand-crafted features, lexicons, and grammars.

Meanwhile, recurrent neural networks (RNNs) have made swift inroads into many structured prediction tasks in NLP, including machine translation (Sutskever et al., 2014; Bahdanau et al., 2014) and syntactic parsing (Vinyals et al., 2015b; Dyer et al., 2015). Because RNNs make very few domain-specific assumptions, they have the potential to succeed at a wide variety of tasks with minimal feature engineering. However, this flexibility also puts RNNs at a disadvantage compared to standard semantic parsers, which can generalize naturally by leveraging their built-in awareness of logical compositionality.

In this paper, we introduce data recombination, a generic framework for declaratively inject-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 12–22,
Berlin, Germany, August 7-12, 2016.

GEO
x: "*what is the population of iowa ?*"
y:
```
_answer ( NV , (
  _population ( NV , V1 ) , _const (
    V0 , _stateid ( iowa ) ) ) )
```
ATIS
x: "*can you list all flights from chicago to milwaukee*"
y:
```
( _lambda $0 e ( _and
  ( _flight $0 )
  ( _from $0 chicago :  _ci )
  ( _to $0 milwaukee :  _ci ) ) )
```
Overnight
x: "*when is the weekly standup*"
y:
```
( call listValue ( call
    getProperty meeting.weekly_standup
    ( string start_time ) ) )
```

Figure 2: One example from each of our domains. We tokenize logical forms as shown, thereby casting semantic parsing as a sequence-to-sequence task.

ing prior knowledge into a domain-general structured prediction model. In data recombination, prior knowledge about a task is used to build a high-precision generative model that expands the empirical distribution by allowing fragments of different examples to be combined in particular ways. Samples from this generative model are then used to train a domain-general model. In the case of semantic parsing, we construct a generative model by inducing a synchronous context-free grammar (SCFG), creating new examples such as those shown in Figure 1; our domain-general model is a sequence-to-sequence RNN with a novel attention-based copying mechanism. Data recombination boosts the accuracy of our RNN model on three semantic parsing datasets. On the GEO dataset, data recombination improves test accuracy by 4.3 percentage points over our baseline RNN, leading to new state-of-the-art results for models that do not use a seed lexicon for predicates.

2 Problem statement

We cast semantic parsing as a sequence-to-sequence task. The input utterance x is a sequence of words $x_1, \ldots, x_m \in \mathcal{V}^{(\text{in})}$, the input vocabulary; similarly, the output logical form y is a sequence of tokens $y_1, \ldots, y_n \in \mathcal{V}^{(\text{out})}$, the output vocabulary. A linear sequence of tokens might appear to lose the hierarchical structure of a logical form, but there is precedent for this choice: Vinyals et al. (2015b) showed that an RNN can reliably predict tree-structured outputs in a linear fashion.

We evaluate our system on three existing semantic parsing datasets. Figure 2 shows sample input-output pairs from each of these datasets.

- **GeoQuery** (GEO) contains natural language questions about US geography paired with corresponding Prolog database queries. We use the standard split of 600 training examples and 280 test examples introduced by Zettlemoyer and Collins (2005). We preprocess the logical forms to De Brujin index notation to standardize variable naming.
- **ATIS** (ATIS) contains natural language queries for a flights database paired with corresponding database queries written in lambda calculus. We train on 4473 examples and evaluate on the 448 test examples used by Zettlemoyer and Collins (2007).
- **Overnight** (OVERNIGHT) contains logical forms paired with natural language paraphrases across eight varied subdomains. Wang et al. (2015) constructed the dataset by generating all possible logical forms up to some depth threshold, then getting multiple natural language paraphrases for each logical form from workers on Amazon Mechanical Turk. We evaluate on the same train/test splits as Wang et al. (2015).

In this paper, we only explore learning from logical forms. In the last few years, there has an emergence of semantic parsers learned from denotations (Clarke et al., 2010; Liang et al., 2011; Berant et al., 2013; Artzi and Zettlemoyer, 2013b). While our system cannot directly learn from denotations, it could be used to rerank candidate derivations generated by one of these other systems.

3 Sequence-to-sequence RNN Model

Our sequence-to-sequence RNN model is based on existing attention-based neural machine translation models (Bahdanau et al., 2014; Luong et al., 2015a), but also includes a novel attention-based copying mechanism. Similar copying mechanisms have been explored in parallel by Gu et al. (2016) and Gulcehre et al. (2016).

3.1 Basic Model

Encoder. The encoder converts the input sequence $x_1, \ldots, x_m$ into a sequence of *context-*

sensitive embeddings $b_1, \ldots, b_m$ using a bidirectional RNN (Bahdanau et al., 2014). First, a word embedding function $\phi^{(\text{in})}$ maps each word x_i to a fixed-dimensional vector. These vectors are fed as input to two RNNs: a forward RNN and a backward RNN. The forward RNN starts with an initial hidden state h_0^{F}, and generates a sequence of hidden states $h_1^{\text{F}}, \ldots, h_m^{\text{F}}$ by repeatedly applying the recurrence

$$h_i^{\text{F}} = \text{LSTM}(\phi^{(\text{in})}(x_i), h_{i-1}^{\text{F}}). \quad (1)$$

The recurrence takes the form of an LSTM (Hochreiter and Schmidhuber, 1997). The backward RNN similarly generates hidden states $h_m^{\text{B}}, \ldots, h_1^{\text{B}}$ by processing the input sequence in reverse order. Finally, for each input position i, we define the context-sensitive embedding b_i to be the concatenation of h_i^{F} and h_i^{B}

Decoder. The decoder is an attention-based model (Bahdanau et al., 2014; Luong et al., 2015a) that generates the output sequence $y_1, \ldots, y_n$ one token at a time. At each time step j, it writes y_j based on the current hidden state s_j, then updates the hidden state to s_{j+1} based on s_j and y_j. Formally, the decoder is defined by the following equations:

$$s_1 = \tanh(W^{(s)}[h_m^{\text{F}}, h_1^{\text{B}}]). \quad (2)$$

$$e_{ji} = s_j^\top W^{(a)} b_i. \quad (3)$$

$$\alpha_{ji} = \frac{\exp(e_{ji})}{\sum_{i'=1}^{m} \exp(e_{ji'})}. \quad (4)$$

$$c_j = \sum_{i=1}^{m} \alpha_{ji} b_i. \quad (5)$$

$$P(y_j = w \mid x, y_{1:j-1}) \propto \exp(U_w[s_j, c_j]). \quad (6)$$

$$s_{j+1} = \text{LSTM}([\phi^{(\text{out})}(y_j), c_j], s_j). \quad (7)$$

When not specified, i ranges over $\{1, \ldots, m\}$ and j ranges over $\{1, \ldots, n\}$. Intuitively, the α_{ji}'s define a probability distribution over the input words, describing what words in the input the decoder is focusing on at time j. They are computed from the unnormalized attention scores e_{ji}. The matrices $W^{(s)}$, $W^{(a)}$, and U, as well as the embedding function $\phi^{(\text{out})}$, are parameters of the model.

3.2 Attention-based Copying

In the basic model of the previous section, the next output word y_j is chosen via a simple softmax over all words in the output vocabulary. However, this model has difficulty generalizing to the long tail of entity names commonly found in semantic parsing datasets. Conveniently, entity names in the input often correspond directly to tokens in the output (e.g., "*iowa*" becomes `iowa` in Figure 2).[1]

To capture this intuition, we introduce a new attention-based copying mechanism. At each time step j, the decoder generates one of two types of actions. As before, it can write any word in the output vocabulary. In addition, it can copy any input word x_i directly to the output, where the probability with which we copy x_i is determined by the attention score on x_i. Formally, we define a latent action a_j that is either $\texttt{Write}[w]$ for some $w \in \mathcal{V}^{(\text{out})}$ or $\texttt{Copy}[i]$ for some $i \in \{1, \ldots, m\}$. We then have

$$P(a_j = \texttt{Write}[w] \mid x, y_{1:j-1}) \propto \exp(U_w[s_j, c_j]), \quad (8)$$

$$P(a_j = \texttt{Copy}[i] \mid x, y_{1:j-1}) \propto \exp(e_{ji}). \quad (9)$$

The decoder chooses a_j with a softmax over all these possible actions; y_j is then a deterministic function of a_j and x. During training, we maximize the log-likelihood of y, marginalizing out a.

Attention-based copying can be seen as a combination of a standard softmax output layer of an attention-based model (Bahdanau et al., 2014) and a Pointer Network (Vinyals et al., 2015a); in a Pointer Network, the only way to generate output is to copy a symbol from the input.

4 Data Recombination

4.1 Motivation

The main contribution of this paper is a novel data recombination framework that injects important prior knowledge into our oblivious sequence-to-sequence RNN. In this framework, we induce a high-precision generative model from the training data, then sample from it to generate new training examples. The process of inducing this generative model can leverage any available prior knowledge, which is transmitted through the generated examples to the RNN model. A key advantage of our two-stage approach is that it allows us to declare desired properties of the task which might be hard to capture in the model architecture.

[1] On GEO and ATIS, we make a point not to rely on orthography for non-entities such as "*state*" to `_state`, since this leverages information not available to previous models (Zettlemoyer and Collins, 2005) and is much less language-independent.

Examples
("*what states border texas ?*",
`answer(NV, (state(V0), next_to(V0, NV), const(V0, stateid(texas))))`)
("*what is the highest mountain in ohio ?*",
`answer(NV, highest(V0, (mountain(V0), loc(V0, NV), const(V0, stateid(ohio)))))`)

Rules created by ABSENTITIES
ROOT → ⟨ "*what states border* STATEID *?*",
`answer(NV, (state(V0), next_to(V0, NV), const(V0, stateid(`STATEID`))))`⟩
STATEID → ⟨ "*texas*", `texas` ⟩
ROOT → ⟨ "*what is the highest mountain in* STATEID *?*",
`answer(NV, highest(V0, (mountain(V0), loc(V0, NV),`
`const(V0, stateid(`STATEID`)))))`⟩
STATEID → ⟨"*ohio*", `ohio`⟩

Rules created by ABSWHOLEPHRASES
ROOT → ⟨ "*what states border* STATE *?*", `answer(NV, (state(V0), next_to(V0, NV),` STATE `))`⟩
STATE → ⟨ "*states border texas*", `state(V0), next_to(V0, NV), const(V0, stateid(texas))`⟩
ROOT → ⟨ "*what is the highest mountain in* STATE *?*",
`answer(NV, highest(V0, (mountain(V0), loc(V0, NV),` STATE `)))`⟩

Rules created by CONCAT-2
ROOT → ⟨SENT$_1$ `</s>` SENT$_2$, SENT$_1$ `</s>` SENT$_2$⟩
SENT → ⟨ "*what states border texas ?*",
`answer(NV, (state(V0), next_to(V0, NV), const(V0, stateid(texas))))` ⟩
SENT → ⟨ "*what is the highest mountain in ohio ?*",
`answer(NV, highest(V0, (mountain(V0), loc(V0, NV), const(V0, stateid(ohio)))))` ⟩

Figure 3: Various grammar induction strategies illustrated on GEO. Each strategy converts the rules of an input grammar into rules of an output grammar. This figure shows the base case where the input grammar has rules ROOT → $\langle x, y \rangle$ for each (x, y) pair in the training dataset.

Our approach generalizes data augmentation, which is commonly employed to inject prior knowledge into a model. Data augmentation techniques focus on modeling invariances—transformations like translating an image or adding noise that alter the inputs x, but do not change the output y. These techniques have proven effective in areas like computer vision (Krizhevsky et al., 2012) and speech recognition (Jaitly and Hinton, 2013).

In semantic parsing, however, we would like to capture more than just invariance properties. Consider an example with the utterance "*what states border texas ?*". Given this example, it should be easy to generalize to questions where "*texas*" is replaced by the name of any other state: simply replace the mention of Texas in the logical form with the name of the new state. Underlying this phenomenon is a strong conditional independence principle: the meaning of the rest of the sentence is independent of the name of the state in question. Standard data augmentation is not sufficient to model such phenomena: instead of holding y fixed, we would like to apply simultaneous transformations to x and y such that the new x still maps to the new y. Data recombination addresses this need.

4.2 General Setting

In the general setting of data recombination, we start with a training set $\mathcal{D}$ of (x, y) pairs, which defines the empirical distribution $\hat{p}(x, y)$. We then fit a generative model $\tilde{p}(x, y)$ to $\hat{p}$ which generalizes beyond the support of $\hat{p}$, for example by splicing together fragments of different examples. We refer to examples in the support of $\tilde{p}$ as *recombinant* examples. Finally, to train our actual model $p_\theta(y \mid x)$, we maximize the expected value of $\log p_\theta(y \mid x)$, where (x, y) is drawn from $\tilde{p}$.

4.3 SCFGs for Semantic Parsing

For semantic parsing, we induce a synchronous context-free grammar (SCFG) to serve as the backbone of our generative model $\tilde{p}$. An SCFG consists of a set of production rules $X \to \langle \alpha, \beta \rangle$, where X is a category (non-terminal), and α and β are sequences of terminal and non-terminal symbols. Any non-terminal symbols in α must be aligned to the same non-terminal symbol in β, and vice versa. Therefore, an SCFG defines a set of joint derivations of aligned pairs of strings. In our case, we use an SCFG to represent joint deriva-

tions of utterances x and logical forms y (which for us is just a sequence of tokens). After we induce an SCFG G from $\mathcal{D}$, the corresponding generative model $\tilde{p}(x, y)$ is the distribution over pairs (x, y) defined by sampling from G, where we choose production rules to apply uniformly at random.

It is instructive to compare our SCFG-based data recombination with WASP (Wong and Mooney, 2006; Wong and Mooney, 2007), which uses an SCFG as the actual semantic parsing model. The grammar induced by WASP must have good coverage in order to generalize to new inputs at test time. WASP also requires the implementation of an efficient algorithm for computing the conditional probability $p(y \mid x)$. In contrast, our SCFG is only used to convey prior knowledge about conditional independence structure, so it only needs to have high precision; our RNN model is responsible for boosting recall over the entire input space. We also only need to forward sample from the SCFG, which is considerably easier to implement than conditional inference.

Below, we examine various strategies for inducing a grammar G from a dataset $\mathcal{D}$. We first encode $\mathcal{D}$ as an initial grammar with rules ROOT $\rightarrow \langle x, y \rangle$ for each $(x, y) \in \mathcal{D}$. Next, we will define each grammar induction strategy as a mapping from an input grammar G_{in} to a new grammar G_{out}. This formulation allows us to compose grammar induction strategies (Section 4.3.4).

4.3.1 Abstracting Entities

Our first grammar induction strategy, ABSENTITIES, simply abstracts entities with their types. We assume that each entity e (e.g., `texas`) has a corresponding type $e.t$ (e.g., `state`), which we infer based on the presence of certain predicates in the logical form (e.g. `stateid`). For each grammar rule $X \rightarrow \langle \alpha, \beta \rangle$ in G_{in}, where α contains a token (e.g., "*texas*") that string matches an entity (e.g., `texas`) in β, we add two rules to G_{out}: (i) a rule where both occurrences are replaced with the type of the entity (e.g., `state`), and (ii) a new rule that maps the type to the entity (e.g., STATEID $\rightarrow \langle$"*texas*", `texas`$\rangle$; we reserve the category name STATE for the next section). Thus, G_{out} generates recombinant examples that fuse most of one example with an entity found in a second example. A concrete example from the GEO domain is given in Figure 3.

4.3.2 Abstracting Whole Phrases

Our second grammar induction strategy, ABSWHOLEPHRASES, abstracts both entities and whole phrases with their types. For each grammar rule $X \rightarrow \langle \alpha, \beta \rangle$ in G_{in}, we add up to two rules to G_{out}. First, if α contains tokens that string match to an entity in β, we replace both occurrences with the type of the entity, similarly to rule (i) from ABSENTITIES. Second, if we can infer that the entire expression β evaluates to a set of a particular type (e.g. `state`) we create a rule that maps the type to $\langle \alpha, \beta \rangle$. In practice, we also use some simple rules to strip question identifiers from α, so that the resulting examples are more natural. Again, refer to Figure 3 for a concrete example.

This strategy works because of a more general conditional independence property: the meaning of any semantically coherent phrase is conditionally independent of the rest of the sentence, the cornerstone of compositional semantics. Note that this assumption is not always correct in general: for example, phenomena like anaphora that involve long-range context dependence violate this assumption. However, this property holds in most existing semantic parsing datasets.

4.3.3 Concatenation

The final grammar induction strategy is a surprisingly simple approach we tried that turns out to work. For any $k \geq 2$, we define the CONCAT-k strategy, which creates two types of rules. First, we create a single rule that has ROOT going to a sequence of k SENT's. Then, for each root-level rule ROOT $\rightarrow \langle \alpha, \beta \rangle$ in G_{in}, we add the rule SENT $\rightarrow \langle \alpha, \beta \rangle$ to G_{out}. See Figure 3 for an example.

Unlike ABSENTITIES and ABSWHOLEPHRASES, concatenation is very general, and can be applied to any sequence transduction problem. Of course, it also does not introduce additional information about compositionality or independence properties present in semantic parsing. However, it does generate harder examples for the attention-based RNN, since the model must learn to attend to the correct parts of the now-longer input sequence. Related work has shown that training a model on more difficult examples can improve generalization, the most canonical case being dropout (Hinton et al., 2012; Wager et al., 2013).

```
function TRAIN(dataset D, number of epochs T,
      number of examples to sample n)
    Induce grammar G from D
    Initialize RNN parameters θ randomly
    for each iteration t = 1, ..., T do
        Compute current learning rate η_t
        Initialize current dataset D_t to D
        for i = 1, ..., n do
            Sample new example (x', y') from G
            Add (x', y') to D_t
        end for
        Shuffle D_t
        for each example (x, y) in D_t do
            θ ← θ + η_t ∇ log p_θ(y | x)
        end for
    end for
end function
```

Figure 4: The training procedure with data recombination. We first induce an SCFG, then sample new recombinant examples from it at each epoch.

4.3.4 Composition

We note that grammar induction strategies can be composed, yielding more complex grammars. Given any two grammar induction strategies f_1 and f_2, the composition $f_1 \circ f_2$ is the grammar induction strategy that takes in G_{in} and returns $f_1(f_2(G_{\text{in}}))$. For the strategies we have defined, we can perform this operation symbolically on the grammar rules, without having to sample from the intermediate grammar $f_2(G_{\text{in}})$.

5 Experiments

We evaluate our system on three domains: GEO, ATIS, and OVERNIGHT. For ATIS, we report logical form exact match accuracy. For GEO and OVERNIGHT, we determine correctness based on denotation match, as in Liang et al. (2011) and Wang et al. (2015), respectively.

5.1 Choice of Grammar Induction Strategy

We note that not all grammar induction strategies make sense for all domains. In particular, we only apply ABSWHOLEPHRASES to GEO and OVERNIGHT. We do not apply ABSWHOLEPHRASES to ATIS, as the dataset has little nesting structure.

5.2 Implementation Details

We tokenize logical forms in a domain-specific manner, based on the syntax of the formal language being used. On GEO and ATIS, we disallow copying of predicate names to ensure a fair comparison to previous work, as string matching between input words and predicate names is not commonly used. We prevent copying by prepending underscores to predicate tokens; see Figure 2 for examples.

On ATIS alone, when doing attention-based copying and data recombination, we leverage an external lexicon that maps natural language phrases (e.g., "*kennedy airport*") to entities (e.g., `jfk:ap`). When we copy a word that is part of a phrase in the lexicon, we write the entity associated with that lexicon entry. When performing data recombination, we identify entity alignments based on matching phrases and entities from the lexicon.

We run all experiments with 200 hidden units and 100-dimensional word vectors. We initialize all parameters uniformly at random within the interval $[-0.1, 0.1]$. We maximize the log-likelihood of the correct logical form using stochastic gradient descent. We train the model for a total of 30 epochs with an initial learning rate of 0.1, and halve the learning rate every 5 epochs, starting after epoch 15. We replace word vectors for words that occur only once in the training set with a universal `<unk>` word vector. Our model is implemented in Theano (Bergstra et al., 2010).

When performing data recombination, we sample a new round of recombinant examples from our grammar at each epoch. We add these examples to the original training dataset, randomly shuffle all examples, and train the model for the epoch. Figure 4 gives pseudocode for this training procedure. One important hyperparameter is how many examples to sample at each epoch: we found that a good rule of thumb is to sample as many recombinant examples as there are examples in the training dataset, so that half of the examples the model sees at each epoch are recombinant.

At test time, we use beam search with beam size 5. We automatically balance missing right parentheses by adding them at the end. On GEO and OVERNIGHT, we then pick the highest-scoring logical form that does not yield an executor error when the corresponding denotation is computed. On ATIS, we just pick the top prediction on the beam.

5.3 Impact of the Copying Mechanism

First, we measure the contribution of the attention-based copying mechanism to the model's overall

	GEO	ATIS	OVERNIGHT
No Copying	74.6	69.9	76.7
With Copying	85.0	76.3	75.8

Table 1: Test accuracy on GEO, ATIS, and OVERNIGHT, both with and without copying. On OVERNIGHT, we average across all eight domains.

	GEO	ATIS
Previous Work		
Zettlemoyer and Collins (2007)		**84.6**
Kwiatkowski et al. (2010)	88.9	
Liang et al. (2011)[2]	91.1	
Kwiatkowski et al. (2011)	88.6	82.8
Poon (2013)		83.5
Zhao and Huang (2015)	88.9	84.2
Our Model		
No Recombination	85.0	76.3
ABSENTITIES	85.4	79.9
ABSWHOLEPHRASES	87.5	
CONCAT-2	84.6	79.0
CONCAT-3		77.5
AWP + AE	88.9	
AE + C2		78.8
AWP + AE + C2	**89.3**	
AE + C3		83.3

Table 2: Test accuracy using different data recombination strategies on GEO and ATIS. AE is ABSENTITIES, AWP is ABSWHOLEPHRASES, C2 is CONCAT-2, and C3 is CONCAT-3.

performance. On each task, we train and evaluate two models: one with the copying mechanism, and one without. Training is done without data recombination. The results are shown in Table 1.

On GEO and ATIS, the copying mechanism helps significantly: it improves test accuracy by 10.4 percentage points on GEO and 6.4 points on ATIS. However, on OVERNIGHT, adding the copying mechanism actually makes our model perform slightly worse. This result is somewhat expected, as the OVERNIGHT dataset contains a very small number of distinct entities. It is also notable that both systems surpass the previous best system on OVERNIGHT by a wide margin.

We choose to use the copying mechanism in all subsequent experiments, as it has a large advantage in realistic settings where there are many distinct entities in the world. The concurrent work of Gu et al. (2016) and Gulcehre et al. (2016), both of whom propose similar copying mechanisms, provides additional evidence for the utility of copying on a wide range of NLP tasks.

5.4 Main Results

For our main results, we train our model with a variety of data recombination strategies on all three datasets. These results are summarized in Tables 2 and 3. We compare our system to the baseline of not using any data recombination, as well as to state-of-the-art systems on all three datasets.

We find that data recombination consistently improves accuracy across the three domains we evaluated on, and that the strongest results come from composing multiple strategies. Combining ABSWHOLEPHRASES, ABSENTITIES, and CONCAT-2 yields a 4.3 percentage point improvement over the baseline without data recombination on GEO, and an average of 1.7 percentage points on OVERNIGHT. In fact, on GEO, we achieve test accuracy of 89.3%, which surpasses the previous state-of-the-art, excluding Liang et al. (2011), which used a seed lexicon for predicates. On ATIS, we experiment with concatenating more than 2 examples, to make up for the fact that we cannot apply ABSWHOLEPHRASES, which generates longer examples. We obtain a test accuracy of 83.3 with ABSENTITIES composed with CONCAT-3, which beats the baseline by 7 percentage points and is competitive with the state-of-the-art.

Data recombination without copying. For completeness, we also investigated the effects of data recombination on the model without attention-based copying. We found that recombination helped significantly on GEO and ATIS, but hurt the model slightly on OVERNIGHT. On GEO, the best data recombination strategy yielded test accuracy of 82.9%, for a gain of 8.3 percentage points over the baseline with no copying and no recombination; on ATIS, data recombination gives test accuracies as high as 74.6%, a 4.7 point gain over the same baseline. However, no data recombination strategy improved average test accuracy on OVERNIGHT; the best one resulted in a 0.3 percentage point decrease in test accuracy. We hypothesize that data recombination helps less on OVERNIGHT in general because the space of possible logical forms is very limited, making it more like a large multiclass classification task. Therefore, it is less important for the model to learn good compositional representations that generalize to new logical forms at test time.

[2]The method of Liang et al. (2011) is not comparable to ours, as they as they used a seed lexicon mapping words to predicates. We explicitly avoid using such prior knowledge in our system.

	BASKETBALL	BLOCKS	CALENDAR	HOUSING	PUBLICATIONS	RECIPES	RESTAURANTS	SOCIAL	Avg.
Previous Work									
Wang et al. (2015)	46.3	41.9	74.4	54.0	59.0	70.8	75.9	48.2	58.8
Our Model									
No Recombination	85.2	58.1	78.0	71.4	76.4	79.6	76.2	81.4	75.8
ABSENTITIES	86.7	60.2	78.0	65.6	73.9	77.3	79.5	81.3	75.3
ABSWHOLEPHRASES	86.7	55.9	79.2	69.8	76.4	77.8	80.7	80.9	75.9
CONCAT-2	84.7	**60.7**	75.6	69.8	74.5	80.1	79.5	80.8	75.7
AWP + AE	85.2	54.1	78.6	67.2	73.9	79.6	**81.9**	**82.1**	75.3
AWP + AE + C2	**87.5**	60.2	**81.0**	**72.5**	**78.3**	**81.0**	79.5	79.6	**77.5**

Table 3: Test accuracy using different data recombination strategies on the OVERNIGHT tasks.

Depth-2 (same length)
x: "*rel:12 of rel:17 of ent:14*"
y: `( _rel:12 ( _rel:17 _ent:14 ) )`
Depth-4 (longer)
x: "*rel:23 of rel:36 of rel:38 of rel:10 of ent:05*"
y: `( _rel:23 ( _rel:36 ( _rel:38 ( _rel:10 _ent:05 ) ) ) )`

Figure 5: A sample of our artificial data.

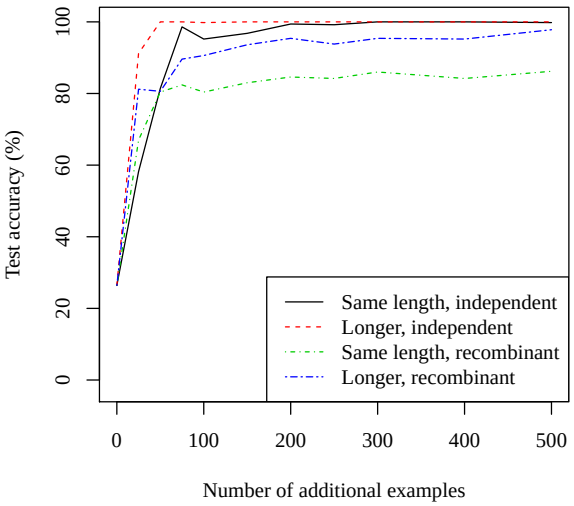

Figure 6: The results of our artificial data experiments. We see that the model learns more from longer examples than from same-length examples.

5.5 Effect of Longer Examples

Interestingly, strategies like ABSWHOLEPHRASES and CONCAT-2 help the model even though the resulting recombinant examples are generally not in the support of the test distribution. In particular, these recombinant examples are on average longer than those in the actual dataset, which makes them harder for the attention-based model. Indeed, for every domain, our best accuracy numbers involved some form of concatenation, and often involved ABSWHOLEPHRASES as well. In comparison, applying ABSENTITIES alone, which generates examples of the same length as those in the original dataset, was generally less effective.

We conducted additional experiments on artificial data to investigate the importance of adding longer, harder examples. We experimented with adding new examples via data recombination, as well as adding new independent examples (e.g. to simulate the acquisition of more training data). We constructed a simple world containing a set of entities and a set of binary relations. For any n, we can generate a set of depth-n examples, which involve the composition of n relations applied to a single entity. Example data points are shown in Figure 5. We train our model on various datasets, then test it on a set of 500 randomly chosen depth-2 examples. The model always has access to a small seed training set of 100 depth-2 examples. We then add one of four types of examples to the training set:

- **Same length, independent**: New randomly chosen depth-2 examples.[3]
- **Longer, independent**: Randomly chosen depth-4 examples.
- **Same length, recombinant**: Depth-2 examples sampled from the grammar induced by applying ABSENTITIES to the seed dataset.
- **Longer, recombinant**: Depth-4 examples sampled from the grammar induced by applying ABSWHOLEPHRASES followed by ABSENTITIES to the seed dataset.

To maintain consistency between the independent and recombinant experiments, we fix the recombinant examples across all epochs, instead of resampling at every epoch. In Figure 6, we plot accuracy on the test set versus the number of additional examples added of each of these four types. As

[3]Technically, these are not completely independent, as we sample these new examples without replacement. The same applies to the longer "independent" examples.

expected, independent examples are more helpful than the recombinant ones, but both help the model improve considerably. In addition, we see that even though the test dataset only has short examples, adding longer examples helps the model more than adding shorter ones, in both the independent and recombinant cases. These results underscore the importance training on longer, harder examples.

6 Discussion

In this paper, we have presented a novel framework we term data recombination, in which we generate new training examples from a high-precision generative model induced from the original training dataset. We have demonstrated its effectiveness in improving the accuracy of a sequence-to-sequence RNN model on three semantic parsing datasets, using a synchronous context-free grammar as our generative model.

There has been growing interest in applying neural networks to semantic parsing and related tasks. Dong and Lapata (2016) concurrently developed an attention-based RNN model for semantic parsing, although they did not use data recombination. Grefenstette et al. (2014) proposed a non-recurrent neural model for semantic parsing, though they did not run experiments. Mei et al. (2016) use an RNN model to perform a related task of instruction following.

Our proposed attention-based copying mechanism bears a strong resemblance to two models that were developed independently by other groups. Gu et al. (2016) apply a very similar copying mechanism to text summarization and single-turn dialogue generation. Gulcehre et al. (2016) propose a model that decides at each step whether to write from a "shortlist" vocabulary or copy from the input, and report improvements on machine translation and text summarization. Another piece of related work is Luong et al. (2015b), who train a neural machine translation system to copy rare words, relying on an external system to generate alignments.

Prior work has explored using paraphrasing for data augmentation on NLP tasks. Zhang et al. (2015) augment their data by swapping out words for synonyms from WordNet. Wang and Yang (2015) use a similar strategy, but identify similar words and phrases based on cosine distance between vector space embeddings. Unlike our data recombination strategies, these techniques only change inputs x, while keeping the labels y fixed. Additionally, these paraphrasing-based transformations can be described in terms of grammar induction, so they can be incorporated into our framework.

In data recombination, data generated by a high-precision generative model is used to train a second, domain-general model. Generative oversampling (Liu et al., 2007) learns a generative model in a multiclass classification setting, then uses it to generate additional examples from rare classes in order to combat label imbalance. Uptraining (Petrov et al., 2010) uses data labeled by an accurate but slow model to train a computationally cheaper second model. Vinyals et al. (2015b) generate a large dataset of constituency parse trees by taking sentences that multiple existing systems parse in the same way, and train a neural model on this dataset.

Some of our induced grammars generate examples that are not in the test distribution, but nonetheless aid in generalization. Related work has also explored the idea of training on altered or out-of-domain data, often interpreting it as a form of regularization. Dropout training has been shown to be a form of adaptive regularization (Hinton et al., 2012; Wager et al., 2013). Guu et al. (2015) showed that encouraging a knowledge base completion model to handle longer path queries acts as a form of structural regularization.

Language is a blend of crisp regularities and soft relationships. Our work takes RNNs, which excel at modeling soft phenomena, and uses a highly structured tool—synchronous context free grammars—to infuse them with an understanding of crisp structure. We believe this paradigm for simultaneously modeling the soft and hard aspects of language should have broader applicability beyond semantic parsing.

Acknowledgments This work was supported by the NSF Graduate Research Fellowship under Grant No. DGE-114747, and the DARPA Communicating with Computers (CwC) program under ARO prime contract no. W911NF-15-1-0462.

Reproducibility. All code, data, and experiments for this paper are available on the CodaLab platform at `https://worksheets.codalab.org/worksheets/0x50757a37779b485f89012e4ba03b6f4f/`.

References

Y. Artzi and L. Zettlemoyer. 2013a. UW SPF: The University of Washington semantic parsing framework. *arXiv preprint arXiv:1311.3011*.

Y. Artzi and L. Zettlemoyer. 2013b. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics (TACL)*, 1:49–62.

D. Bahdanau, K. Cho, and Y. Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

J. Berant, A. Chou, R. Frostig, and P. Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)*.

J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Python for Scientific Computing Conference*.

J. Clarke, D. Goldwasser, M. Chang, and D. Roth. 2010. Driving semantic parsing from the world's response. In *Computational Natural Language Learning (CoNLL)*, pages 18–27.

L. Dong and M. Lapata. 2016. Language to logical form with neural attention. In *Association for Computational Linguistics (ACL)*.

C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Association for Computational Linguistics (ACL)*.

E. Grefenstette, P. Blunsom, N. de Freitas, and K. M. Hermann. 2014. A deep architecture for semantic parsing. In *ACL Workshop on Semantic Parsing*, pages 22–27.

J. Gu, Z. Lu, H. Li, and V. O. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Association for Computational Linguistics (ACL)*.

C. Gulcehre, S. Ahn, R. Nallapati, B. Zhou, and Y. Bengio. 2016. Pointing the unknown words. In *Association for Computational Linguistics (ACL)*.

K. Guu, J. Miller, and P. Liang. 2015. Traversing knowledge graphs in vector space. In *Empirical Methods in Natural Language Processing (EMNLP)*.

G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

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

N. Jaitly and G. E. Hinton. 2013. Vocal tract length perturbation (vtlp) improves speech recognition. In *International Conference on Machine Learning (ICML)*.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105.

N. Kushman and R. Barzilay. 2013. Using semantic unification to generate regular expressions from natural language. In *Human Language Technology and North American Association for Computational Linguistics (HLT/NAACL)*, pages 826–836.

T. Kwiatkowski, L. Zettlemoyer, S. Goldwater, and M. Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1223–1233.

T. Kwiatkowski, L. Zettlemoyer, S. Goldwater, and M. Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1512–1523.

P. Liang, M. I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *Association for Computational Linguistics (ACL)*, pages 590–599.

A. Liu, J. Ghosh, and C. Martin. 2007. Generative oversampling for mining imbalanced datasets. In *International Conference on Data Mining (DMIN)*.

M. Luong, H. Pham, and C. D. Manning. 2015a. Effective approaches to attention-based neural machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1412–1421.

M. Luong, I. Sutskever, Q. V. Le, O. Vinyals, and W. Zaremba. 2015b. Addressing the rare word problem in neural machine translation. In *Association for Computational Linguistics (ACL)*, pages 11–19.

H. Mei, M. Bansal, and M. R. Walter. 2016. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

S. Petrov, P. Chang, M. Ringgaard, and H. Alshawi. 2010. Uptraining for accurate deterministic question parsing. In *Empirical Methods in Natural Language Processing (EMNLP)*.

H. Poon. 2013. Grounded unsupervised semantic parsing. In *Association for Computational Linguistics (ACL)*.

I. Sutskever, O. Vinyals, and Q. V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112.

O. Vinyals, M. Fortunato, and N. Jaitly. 2015a. Pointer networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2674–2682.

O. Vinyals, L. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. Hinton. 2015b. Grammar as a foreign language. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2755–2763.

S. Wager, S. I. Wang, and P. Liang. 2013. Dropout training as adaptive regularization. In *Advances in Neural Information Processing Systems (NIPS)*.

W. Y. Wang and D. Yang. 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using #petpeeve tweets. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Y. Wang, J. Berant, and P. Liang. 2015. Building a semantic parser overnight. In *Association for Computational Linguistics (ACL)*.

Y. W. Wong and R. J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *North American Association for Computational Linguistics (NAACL)*, pages 439–446.

Y. W. Wong and R. J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Association for Computational Linguistics (ACL)*, pages 960–967.

M. Zelle and R. J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 1050–1055.

L. S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Uncertainty in Artificial Intelligence (UAI)*, pages 658–666.

L. S. Zettlemoyer and M. Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*, pages 678–687.

X. Zhang, J. Zhao, and Y. LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems (NIPS)*.

K. Zhao and L. Huang. 2015. Type-driven incremental semantic parsing with polymorphism. In *North American Association for Computational Linguistics (NAACL)*.