# Learning Out-of-Vocabulary Words in Intelligent Personal Agents

**Avik Ray, Yilin Shen** and **Hongxia Jin**
Samsung Research America, Mountain View, California, USA
{avik.r, yilin.shen, hongxia.jin}@samsung.com

## Abstract

Semantic parsers play a vital role in intelligent agents to convert natural language instructions to an actionable logical form representation. However, after deployment, these parsers suffer from poor accuracy on encountering out-of-vocabulary (OOV) words, or significant accuracy drop on previously supported instructions after retraining. Achieving both goals simultaneously is non-trivial. In this paper, we propose novel neural networks based parsers to learn OOV words; one incorporating a new hybrid paraphrase generation model, and an enhanced sequence-to-sequence model. Extensive experiments on both benchmark and custom datasets show our new parsers achieve significant accuracy gain on OOV words and phrases, and in the meanwhile learn OOV words while maintaining accuracy on previously supported instructions.

## 1 Introduction

Intelligent personal agents (e.g. Alexa, Siri, Bixby) are becoming ubiquitous in the modern world. In order to support a variety of complex natural language instructions/queries posed by humans, these agents use semantic parsers to convert queries to an actionable sequence of logical symbols (aka logical form). Most parsers are trained using supervised machine learning methods on datasets in the form of (query, logical form) tuples. However, such datasets are expensive to collect, resulting in poor vocabulary coverage. For example, a benchmark airline reservation query dataset with 5,410 queries has a vocabulary of just 949 words, compared to a vocabulary of over 100,000 words in an airline review dataset of 41,396 reviews. As a result, once deployed in real world, these parsers perform poorly on queries with OOV words. Therefore, it is critically desirable to design parsers that are (a) able to correctly parse queries even with OOV words; (b) when it fails, it should be able to learn these new OOV words/phrases from paraphrased queries with known words.

Existing semantic parsers mainly fall into two categories: First, traditionally used word lexicon learning based approaches [Zettlemoyer and Collins, 2005; 2007; Kwiatkowski *et al.*, 2011], which require separate lexicon learning algorithm, hand crafted features, or initial seed set of lexicons. The other class is deep neural network based parsers [Jia and Liang, 2016; Dong and Lapata, 2016], which can be trained end to end using the (query, logical form) pairs directly. Commercial personal agents often employ neural semantic parsers due to their scalibility, since these can be trained end to end without hand crafted domain specific grammar/lexicon, and still achieve performance comparable to lexicon learning based approaches. Therefore, in this paper we also focus on developing deep neural network based semantic parsers.

Unfortunately, neural parsers suffer from poor accuracy to OOV words. While it may be possible to learn queries with OOV words by retraining the parser using some paraphrased query with known words, obtained via user intervention [Azaria *et al.*, 2016]. However, in order to maintain the original test accuracy, it usually requires retraining the whole model from scratch for every new added word, which is not scalable. As we show in our experiments, a simple model fine-tuning could lead to significant drop in test accuracy, since optimal model parameters can change significantly with the addition of new words. Therefore, it is crucial to design robust semantic parsers which can both parse queries with out-of-vocabulary words/phrases under different contexts, as well as maintain accuracy on previously correctly parsed queries after model fine-tuning. To this end, in this paper, we design a novel hybrid model which meets both of the above two goals. Our main contributions are as follows.

1. We develop new sequence-to-sequence based neural network parsers, and modifications to existing models which are more robust in tackling queries with out-of-vocabulary words and phrases, under different contexts.
2. A new hybrid paraphrase generation based method is proposed which can be easily integrated to any existing parser, and can be used to improve the out-of-vocabulary parsing accuracy, as well as prevent significant accuracy drop after learning OOV words, due to model fine-tuning.
3. We construct new custom paraphrased dataset for the benchmark Geo-queries, Job-queries, and ATIS datasets, in order to validate the out-of-vocabulary and learning performance of our models.
4. In experimental evaluation we show that our models achieve state-of-the-art accuracy on benchmark datasets. Our models also exhibit significant gain in accuracy to out-of-vocabulary words/phrases over baseline parsers.

### 1.1 Related Work

In this section we discuss prior lines of work most related to this paper. Mapping natural language queries to logical form, also called semantic parsing, has a rich history staring from the early work of [Bobrow, 1964]. Recently the literature has focused on more supervised approaches where parsing rules, grammar, and parameters are automatically learned using labeled examples or associated data. Several approaches have been studied in this regard; using inductive logic programming [Zelle and Mooney, 1996], machine translation [Wong and Mooney, 2007], probabilistic CCG/CFG grammar based parsers [Zettlemoyer and Collins, 2005; 2007; Kwiatkowski *et al.*, 2011; Artzi and Zettlemoyer, 2013; Krishnamurthy, 2016], neural networks [Jia and Liang, 2016; Dong and Lapata, 2016].

Encoder–decoder and sequence-to-sequence based deep neural network architectures have received considerable attention in the recent past due to its success in many different NLP tasks such as machine translation [Cho *et al.*, 2014; Sutskever *et al.*, 2014], parsing [Vinyals *et al.*, 2015b; Jia and Liang, 2016; Dong and Lapata, 2016], automatic text generation [Rush *et al.*, 2015], image captioning [Karpathy and Fei-Fei, 2017], paraphrasing [Prakash *et al.*, 2016]. Many techniques have been proposed to improve the performance of basic sequence-to-sequence architecture by allowing more encoder side information during decoding e.g. using attention [Bahdanau *et al.*, 2015], neural checklists [Kiddon *et al.*, 2016], pointer networks [Vinyals *et al.*, 2015a].

Tackling out-of-vocabulary and rare words is a fundamental problem in different natural language processing/understanding tasks. It has particularly received attention in the problem of machine translation and many different solutions have been proposed [Luong *et al.*, 2015; Luong and Manning, 2016]. The problem of learning new command words have also been studied in CCG based parsers [Thomason *et al.*, 2015; Azaria *et al.*, 2016].

In this paper we study the problem of learning both out-of-vocabulary words and phrases in neural semantic parsers based on sequence-to-sequence neural architecture.

## 2 Problem Definition

In this section we formally define our problem. Any natural language query $\mathbf{q}$ can be expressed as a sequence of words $\mathbf{q} = (w_1, \ldots, w_n)$ where the words $w_i \in \mathcal{V}$, a vocabulary. In semantic parsing we build a parser $\mathcal{P}$ to convert the query $\mathbf{q}$ to its formal meaning representation which can be expressed via a logical form $\mathbf{l}(\mathbf{q}) = (l_1, \ldots, l_m)$, containing logical expression tokens $l_j \in \mathcal{L}$. In the supervised approach we train the parser $\mathcal{P}$ over a training set $T = \{\mathbf{q}_i, \mathbf{l}(\mathbf{q}_i)\}_{i=1}^N$ of (query, logical form) tuples. We would like to build a parser $\mathcal{P}$ which can accomplish two tasks as follows. Suppose a new query $\mathbf{p}^*$ is presented having out-of-vocabulary words or phrases.

**Task 1:** We require parser $\mathcal{P}$ to be robust to out-of-vocabulary words, so that if the new words and phrases in $\mathbf{p}^*$ are semantically close to some words in training vocabulary $\mathcal{V}_T$, it should still be able to find its correct logical form.

In cases when $\mathcal{P}$ cannot parse $\mathbf{p}^*$ correctly, suppose we are able to collect a paraphrased query $\mathbf{q}^*$ of $\mathbf{p}^*$, from possible interactions/dialog with the user [Azaria *et al.*, 2016], which can be correctly parsed to $\mathbf{l}(\mathbf{q}^*)$. Our second task follows.

**Task 2:** Build a new parser $\mathcal{P}'$ by fine-tuning $\mathcal{P}$ using both the paraphrased sample $(\mathbf{p}^*, \mathbf{l}(\mathbf{q}^*))$, and the base training set $T$, such that (a) $\mathcal{P}'$ can correctly map query $\mathbf{p}^*$ to $\mathbf{l}(\mathbf{q}^*)$, and (b) $\mathcal{P}'$ also should be able to correctly understand the new words/phrases when these are used subsequently in different contexts. In essence we want $\mathcal{P}'$ to comprehend the "meaning" of these unknown words and phrases, not just remember them in context of the query $\mathbf{p}^*$.

## 3 Background: Sequence-to-Sequence Models

First we will briefly review the basic sequence-to-sequence based neural network which was recently proposed in [Sutskever *et al.*, 2014] and since has been shown to perform well in many NLP tasks.

Sequence-to-sequence neural networks are useful for modeling systems where both the input and output can be represented as sequences. In our problem the input sequence $(x_1, \ldots, x_n)$ is the query, and the output sequence $(y_1, \ldots, y_m)$ is the logical form expression. A basic sequence-to-sequence neural network consists of an encoder, and a decoder, where each of them can be made up of multiple stacked long short-term memory (LSTM) units. Let $\{h_t\}_{t=1}^n$, $\{h'_t\}_{t=1}^m$ be the encoder and decoder hidden states respectively, and $c$ be the context vector. In attention based models the output $y_t$ is predicted as, $p(y_t = y|y_{<t}) = softmax(W_o a'_t)\mathbf{1}_y$, where $a'_t = \tanh(W_a a_t + W_d h'_t)$, and $a_t$ is the additional encoder attention context vector. A sequence-to-sequence model is trained by maximizing the likelihood function, simplified under the model, as follows.

$$p(y_1, .., y_m|x_1, .., x_n) = \Pi_{t=1}^m p(y_t|y_1, .., y_{t-1}, c) \qquad (1)$$

### 3.1 Sequence-to-Sequence/Tree Parsers

Recently both [Dong and Lapata, 2016], and [Jia and Liang, 2016] showed that the basic sequence-to-sequence with attention model, as discussed in previous section, can also be used for parsing natural language queries to logical form expressions. In some cases when the output logical form can be expressed in a tree structure, e.g. when the logical forms are in $\lambda$-calculus notation, the regular sequence-to-sequence with attention layer may still struggle to learn the tree (or parenthesis) structure of the output. Hence [Dong and Lapata, 2016] proposed another sequence-to-tree model where the decoder performs alternate sequence and tree decoding using intermediate non-terminal tokens.

**Argument replacement:** Logical form tokens typically contains predicates/functions, variables, and constants (or slots). Consider the following query and its logical form from Geo-queries dataset: ("what is the largest city in California?",`(argmax (λ $0 (and (city $0) (loc $0 california:s))) (λ $1 (size $1)))`). Here *california:s* is a constant. [Dong and Lapata, 2016] also proposed an argument replacement step where logical form constants and their corresponding substring in the input query (*california:s* and *California* in the above example) are replaced by special **argument tokens**, before they are fed to the neural parsers. Dong et al. explain that by replacing the wide variability of arguments in the data by few argument tokens, the neural parsers can better learn the semantic structure of the sentence improving accuracy of the basic sequence-to-sequence/tree parser by more than 10%. After argument replacement the above example becomes (“`what is the largest city in` $s0$?”, `(argmax (λ $0 (and (city $0) (loc $0 s0))) (λ $1 (size $1)))`), $s0$ being the argument token. We also perform this argument replacement step in all our models.

## 4 Our Robust Neural Parsers

In this section we describe our new neural models in detail. First, we propose a first-cut solution by modifying existing parsers which improve their out-of-vocabulary accuracy, referred as *Robust Sequence-to-Sequence/Tree models*. Next, we propose a new standalone sequence-to-sequence architecture called *Argument Transfer model*. Third, we propose a novel hybrid *Paraphrase Generation model* which can improve both test accuracy and out-of-vocabulary performance of any existing probabilistic parser.

### 4.1 Robust Sequence-to-Sequence/Tree Models

A major reason of poor performance of sequence-to-sequence parsers to queries with OOV words is the absence of representative embedding of OOV words in the input embedding layer. Usually they are replaced by an unknown token $\langle U \rangle$, whose embedding is initialized at random and learned during training. Instead, embedding of each OOV word can be replaced by its pre-trained word embedding vector [Mikolov *et al.*, 2013; Pennington *et al.*, 2014], if one exists. While this simple modification improves the out-of-vocabulary performance of these models, they still suffer from overfitting and can have poor accuracy upon retraining, as shown in Section 5. Our new models described next mitigates these problems.

### 4.2 Argument Transfer Model

Sequence based models trained using pre-trained word embeddings can still struggle to interpret the argument tokens correctly in the logical form. These argument tokens, present in both query and logical form, are essential for achieving good parsing accuracy [Dong and Lapata, 2016]. Since pre-trained embeddings do not have the embedding corresponding to these custom argument tokens, they are also initialized at random, making their learning difficult. We also observe that the models are able to reliably identify the position of these tokens, but not their exact identity. While queries with single argument can be easily post processed to remove such errors, this is not possible for queries with multiple input arguments. This motivates our argument transfer model.

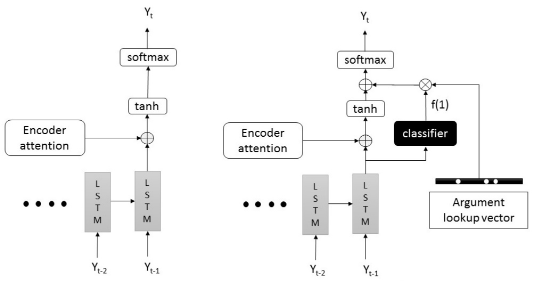
**Model:** The key intuition behind this model is as follows. Suppose we maintain a binary lookup vector identifying which arguments are present at the input query, which can be built during argument replacement step. We can then directly use this vector to restrict our choice of possible arguments in the final softmax layer at the output. Since we have a fixed logical form vocabulary $\mathcal{L}$, the projected attention output $u_t = W_o a'_t$ before the softmax is of dimension $|\mathcal{L}|$. Note that since softmax just performs a non-linear rescaling, essentially each coordinate $i$ of $u_t$ directly measures the contribution to the belief that the output symbol is that corresponding to the dimension $i$. Let $I_A$ be the index set of the output corresponding to the set of argument tokens $A \subset \mathcal{V}$, also let $I_A(\mathbf{q}) \subseteq I_A$ be the subset of the argument indices corresponding to the arguments present in query $\mathbf{q}$. In order to influence the output belief we maintain a lookup vector $v_c \in \{0,1\}^{|\mathcal{L}|}$; where $v_c(i) = 1$ when $i \in I_A(\mathbf{q})$, $v_c(i) = 0$ otherwise. We also train a neural classifier based on the output decoder hidden state $h'_t$ in order to understand when the decoder will generate an argument token, helping the model decide when to use the lookup vector. This is done as follows.

$$f_t = softmax(Ph'_t), \;\; t = 1, \ldots, m \qquad (2)$$

where $f_t \in \mathbb{R}^2$, and $P \in \mathbb{R}^{2\times d}$ is a suitable projection matrix learned during training. Suppose we train the model such that the first index $f_t(1)$ indicates the likelihood that the output token is an argument. We make the final prediction as:

$$p(y_t = y | y_{<t}) = softmax(W_o a'_t + \gamma f_t(1) v_c)\mathbf{1}_y \qquad (3)$$

where $\gamma$ denotes a special temperature hyper-parameter. In essence when the output is an argument, the classifier in (2) will have a value close to 1 in the first index $f_t(1)$. Therefore the lookup vector $v_c$ will directly influence the belief of the output symbol contributing high values $\gamma f_t(1)$ to indices $i \in I_A(\mathbf{q})$, and zero elsewhere. Hence only the belief of the input argument tokens are enhanced at the output. Figure 1 illustrates this decoder architecture of our new model.



(a) sequence-to-sequence + attention decoder (b) argument transfer decoder

Figure 1: Figure (b) showing the decoder architecture of our new argument transfer model versus (a) normal sequence-to-sequence with attention model.

**Training:** In order to jointly learn all parameters of the above argument transfer model we need to add an extra penalty term $\|f_t - f_t^*\|^2$ to our negative log-likelihood objective, where $f_t^*$ represents the ground-truth state of the classifier. As ground-truth we can assign $f_t^* = [1,0]$, for those time instants $t$ having argument token at the output logical form, and $f_t^* = [0,1]$ otherwise.

Sequence-to-sequence models which consider extra signal from the input sequence to modify the output token generation have been proposed recently in various context e.g. for recipe generation [Kiddon *et al.*, 2016], optimization [Vinyals *et al.*, 2015a], semantic parsing [Jia and Liang, 2016]. However, our argument transfer model is architecturally different from all the aforementioned models. In semantic parsing, the softmax output of copy-pointer networks of [Jia and Liang, 2016] predicts token over a combination of input and output vocabulary. In contrast our softmax layer still predicts token over just the output vocabulary, only selectively increasing likelihood of input argument tokens. This explicit transfer of arguments in our model compared to indirect copy mechanism in [Jia and Liang, 2016] considerably improves parsing accuracy on benchmark datasets, as shown in Section 5.

### 4.3 Paraphrase Generation Model

Although neural parsers can be retrained to learn new out-of-vocabulary words using paraphrased examples, it often reduces its accuracy on the original test set, as we will see in experiments. Such accuracy drop is not desirable in commercial personal agents where reliability is of major concern. In other cases one may not have direct access to modify an existing and proprietary parser (e.g. in Alexa or Siri). This motivates our hybrid paraphrase generation model.

Assume there exists a base parser $\mathcal{P}_{base}$ which can parse queries within a training vocabulary $\mathcal{V}_T$, moreover we cannot change/retrain the parser $\mathcal{P}_{base}$. The main idea behind this technique is the following: instead of making the parser $\mathcal{P}_{base}$ robust to unseen words, we can instead build a paraphrase generator $\mathcal{P}_{para}$ which can convert a new query $\mathbf{q}$ to a more familiar paraphrased example with words within the vocabulary $\mathcal{V}_T$, which then can be used as an input to the existing parser $\mathcal{P}_{base}$. The paraphrase generator essentially acts as a translator between the user and $\mathcal{P}_{base}$.

**Algorithm:** In order to prevent the combined model from negatively impacting the test accuracy, we want to be able to determine which queries should be sent to $\mathcal{P}_{para}$ for paraphrasing. We do this using the following algorithm. Assume for each query $\mathbf{q}$, the parser $\mathcal{P}_{base}$ can give a confidence score $s(\mathbf{q})$ to the quality of the parse. In our neural network parsers this can be naturally obtained by computing the likelihood of the output sequence (equation (1)). Next, if the score $s(\mathbf{q})$ is greater than a threshold $\tau$, the parser will output the logical form. If the score $s(\mathbf{q}) \leq \tau$, then it is sent to $\mathcal{P}_{para}$ to form a paraphrased query $\mathbf{q}'$, which then can be parsed again by $\mathcal{P}_{base}$. Such confidence scores can also be obtained in any probabilistic parsers like PCCG [Zettlemoyer and Collins, 2005; 2007], hence can be integrated with any of them. Figure 2 illustrates our algorithm.

**Paraphrase generator:** Many paraphrase generation techniques have been studied in literature e.g. [Zhao *et al.*, 2009; Prakash *et al.*, 2016] which can be used for $\mathcal{P}_{para}$. We implement a sequence-to-sequence (with attention) based paraphrase generator, and use it with our argument transfer parser as $\mathcal{P}_{base}$. We train $\mathcal{P}_{para}$ as follows. We desire $\mathcal{P}_{para}$ to have the property that when query $\mathbf{q}$ has words in the training vocabulary $\mathcal{V}_T$, it may not paraphrase it, and when there are out-of-vocabulary words in $\mathbf{q}$ it should be replaced with most similar words in $\mathcal{V}_T$. We restrict the decoder output of our paraphrase generator to $\mathcal{V}_T$, and the encoder input uses pretrained embedding for initialization. First we train $\mathcal{P}_{para}$ as an autoencoder using a training set $T' = \{(\mathbf{q}_i, \mathbf{q}_i)\}_{\forall \mathbf{q}_i \in T}$. Next, when we obtain new paraphrased examples $(\mathbf{p}^*, \mathbf{q}^*)$ we add this example to $T'$ and retrain $\mathcal{P}_{para}$.

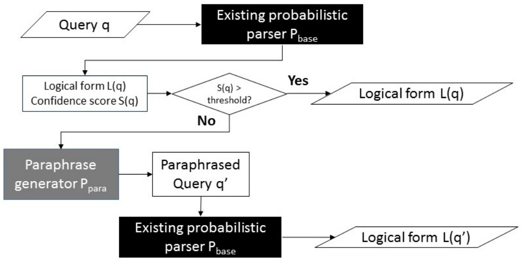


Figure 2: Flowchart illustrating our paraphrase generation model for semantic parsing.

Using paraphrases to improve NLP systems have been studied recently e.g. in QA [Dong *et al.*, 2017], semantic parsing [Berant and Liang, 2014]. Our paraphrase model is unique since our neural generator produces paraphrases having words within the training vocabulary of the base parser, unlike other models. This enables us to improve any existing trained and deployed parser without having to retrain them. In ParaSampre [Berant and Liang, 2014] multiple paraphrases are generated from candidate logical forms and side-information from a KB. The final logical form is selected by measuring similarity between the query and the candidate paraphrases. This is different from our model where we generate paraphrase more directly from the query without any additional side-information. Although we show the applicability of the paraphrase generation model in semantic parsing, this general technique of paraphrasing an input when its output has a low likelihood (hence possibly erroneous), can also be applied in neural machine translation to boost accuracy.

### 4.4 Learning OOV Words

To accomplish our second task of learning new semantically different OOV words/phrases, we assume we are provided with paraphrased examples in the form $(\mathbf{p}^*, \mathbf{q}^*)$, where paraphrased query $\mathbf{q}^*$ has words within the training vocabulary $\mathcal{V}_T$, and can be parsed to $\mathbf{l}(\mathbf{q}^*)$. We add these new examples to the existing training set, and then update the neural network models using fine-tuning. In the paraphrase generation model we retrain $\mathcal{P}_{para}$ but not $\mathcal{P}_{base}$. For fine-tuning, we start with the base parser $\mathcal{P}$, then resume its training starting with a learning rate of the same order as the final learning rate obtained during training. We also limit the number of retraining epochs to that used in training, and stop early if we achieve the final training loss. This considerably reduces retraining time compared to retraining models from scratch.

## 5 Experiments

In this section we describe our main experimental results. The main goals of our evaluation are to determine; (a) how robustly the various neural parsers perform when it encounters

| Original dataset | Original query | Paraphrased query | Logical form | New dataset |
|---|---|---|---|---|
| ATIS | list all flights departing from $ap0$ | list all flights taking off from $ap0$ | ($\lambda$ \$0 (and (flight \$0) (from \$0 $ap0$))) | PARA-ATIS |
| ATIS | i need a flight from $ci0$ to $ci1$ | i require a flight from $ci0$ to $ci1$ | ($\lambda$ \$0 (and (flight \$0) (from \$0 $ci0$) (to \$0 $ci1$))) | PARA-ATIS |
| GEO | how many big cities are in $s0$ | how many large cities are in $s0$ | (count ($\lambda$ \$0 (and (major \$0) (city \$0) (loc \$0 $s0$)))) | PARA-GEO |
| GEO | which state has the highest elevation | which state has the highest natural elevation | (argmax ($\lambda$ \$0 (state \$0)) ($\lambda$ \$1 (elevation \$1))) | PARA-GEO |
| JOB | list job requiring $degid0$ using $languageid0$ | list job requiring $degid0$ applying $languageid0$ | job(ANS), req–deg (ANS,$degid0$), language (ANS,$languageid0$) | PARA-JOB |
| JOB | find all $areaid0$ administrative job in $locid0$ | get all $areaid0$ administrative job in $locid0$ | area(ANS,$areaid0$), job(ANS), loc(ANS,$locid0$) | PARA-JOB |

Table 1: Table showing how our paraphrased datasets were constructed from the original datasets. We replace the underlined words in the original queries with synonymous out-of-vocabulary words and phrases.

out-of-vocabulary words/phrases (b) how well are they able to learn OOV words/phrases after retraining.

## 5.1 Datasets and Methodology

We consider three benchmark semantic parsing datasets as our base datasets. First dataset is the *geographical queries* dataset (GEO) having 880 queries. The second dataset is the *job queries* dataset (JOB) with 640 queries. The third *airline queries* dataset (ATIS) contain overall 5,410 queries (4,480 training, 480 validation, 450 test). In order to evaluate the efficacy of our models we have generated custom paraphrased dataset for every base dataset. We refer them as **PARA-GEO**, **PARA-JOB**, and **PARA-ATIS** datasets in this paper. These paraphrased datasets were generated as follows. We consider each word $w$ in the base dataset queries and find a set of synonymous words and phrases $Syn(w)$ using Wordnet, having the same part-of-speech (POS) tag as in the original queries. For a given train–test/dev data split, let $\{T_t(w), T_d(w)\}$ be the subset of {training, test/dev} queries respectively having the word $w$. We then replace all instances of the word $w$ in both $\{T_t(w), T_d(w)\}$ with each synonym in $s \in Syn(w)$. The datasets were further pruned by a human expert to remove erroneous paraphrases. This generates paraphrased train–test/dev sets $\{T_t(w, s), T_d(w, s)\}$. Therefore our final paraphrased dataset can be represented in the form $\{w, s, T_t(w, s), T_d(w, s)\}$ for each $w \in \mathcal{V}_T, s \in Syn(w) \cap \mathcal{V}_T^c$. Our final PARA-GEO dataset contains 180 (word, synonym) pairs and 5,783 paraphrased queries; PARA-JOB dataset contains 293 (word, synonym) pairs and 7,418 paraphrased queries; the PARA-ATIS dataset contains 161 (word, synonym) pairs and 13,501 paraphrased queries. Table 1 presents example paraphrases from our datasets.

| Model | Test accuracy | | |
|---|---|---|---|
| | **GEO** | **JOB** | **ATIS** |
| $\lambda$-WASP [Wong et al. 2007] | 86.6 | – | – |
| Online CCG [Zettlemoyer et al. 2007] | – | – | 84.6 |
| Seq-to-seq + attn + copy [Jia et al. 2016] | – | – | 83.3 |
| Character seq-to-seq + attn | 77.39 | 83.28 | 84.15 |
| Seq-to-seq + attn [Dong et al. 2016] | 84.89 | 84.69 | 84.2 |
| Seq-to-tree + attn [Dong et al. 2016] | 86.14 | 87.50 | 84.6 |
| Robust seq-to-seq + attn | 85.23 | 87.03 | 85.04 |
| Robust seq-to-tree + attn | 86.36 | **88.59** | 82.59 |
| Argument transfer | 88.18 | **88.59** | **85.27** |
| Paraphrase + arg. transfer | **88.3** | 88.44 | **85.27** |

Table 2: Percentage test accuracy of all models on the GEO, JOB, and ATIS datasets. For GEO and JOB we report 10 fold average[1], while for ATIS we report accuracy for the standard train-test split.

**Methodology:** To validate our models we perform three sets of experiment. Our accuracy metric computes the percentage of test queries which were parsed to their correct logical forms. **1.** First we train the parser $\mathcal{P}$ on the benchmark training set and measure the parsing accuracy (called **test accuracy**) on the test set. **2.** Next we apply parser $\mathcal{P}$ on the test paraphrased examples $T_d(w, s)$ for every word synonym pair $(w, s)$ in the paraphrased dataset, and note the accuracy (referred as **out-of-vocab/OOV accuracy**). We also measure the original parsing accuracy (termed as **reference accuracy**) on the subset of examples $T_d(w)$ in test set from which examples $T_d(w, s)$ were constructed. This enables us to quantify the drop in accuracy caused by out-of-vocabulary words and phrases. **3.** Finally, for the word synonym pairs $(w, s)$ for which the OOV accuracy is less than one, we retrain the model using few paraphrased training examples from $T_t(w, s)$ to obtain the fine-tuned parser $\mathcal{P}'$. We then measure the accuracy of $\mathcal{P}'$ on both the paraphrase test set $T_d(w, s)$ (referred as **regained accuracy**), and the original test set (called **retrained test accuracy**). This allows us to quantify the performance of learning OOV words after retraining.

For GEO and JOB datasets we compute 10 fold cross validation accuracy, since due to their smaller size we observe considerable accuracy variation depending on the particular split, and parameter initialization. For ATIS dataset we compute the test accuracy on the standard split used in [Zettlemoyer and Collins, 2007; Dong and Lapata, 2016]. For retraining, given a $(w, s)$ pair, we always limit the number of new paraphrased training examples from $T_t(w, s)$ to 5, but test the retrained accuracy over the entire test set $T_d(w, s)$.

**Implementation:** We implement all our neural network models using Torch 7 tool. We use the original software made available by [Dong and Lapata, 2016] to implement the baseline sequence-to-sequence/tree models. We adjust hyper-parameters over a separate validation set. We use pretrained GloVe embeddings [Pennington *et al.*, 2014] for all our models. We choose the LSTM hidden state dimension $d \in \{100, 200, 300\}$, and dropout rate in $\{.5, .4, .3, .2\}$. In argument transfer model we take $\gamma \in [4, 6]$. For paraphrase generation model we choose threshold $\tau \in [1, 0.9]$. RMSProp was used as the optimization algorithm.

[1] Note that, the accuracy of the baseline method is lower than that reportend in [Dong and Lapata, 2016] since we compute a 10 fold average (for GEO and JOB), unlike Dong et al. who evaluate accuracy only for the standard train-test split. We further elaborate our methodology in Section 5.1

| Model | PARA-GEO | | | PARA-JOB | | | PARA-ATIS | | |
|---|---|---|---|---|---|---|---|---|---|
| | Reference accuracy | OOV accuracy | Regained accuracy | Reference accuracy | OOV accuracy | Regained accuracy | Reference accuracy | OOV accuracy | Regained accuracy |
| Character seq-to-seq + attn | 59.87 | 41.2 | 46.46 | 81.72 | 71.72 | 80.56 | 66.65 | 55.61 | 58.17 |
| Seq-to-seq + attn [Dong et al. 16] | 76.98 | 52.42 | 57.72 | 84.25 | 76.55 | 83.2 | 70.67 | 53.03 | 56.18 |
| Seq-to-tree + attn [Dong et al. 16] | 75.95 | 53.6 | 59.36 | 87.78 | 83.99 | 86.61 | 61.6 | 46.68 | 47.77 |
| Robust seq-to-seq + attn | 76.08 | 61 | 74.37 | 85.88 | 80.92 | 86.32 | 71.83 | 61.24 | **71.76** |
| Robust seq-to-tree + attn | 79.33 | 62.78 | **77.22** | 88.26 | 83.46 | **89.6** | 63.68 | 49.93 | 59.3 |
| Argument transfer | 80.15 | 62.78 | 75.55 | 89.3 | 83.77 | 88.87 | 75.43 | 57.12 | 69.84 |
| Paraphrase + arg. transfer | 80.17 | **65.61** | 70.17 | 88.92 | **84.81** | 85.99 | 75.43 | **63.5** | 65.62 |

Table 3: Table comparing the percentage reference accuracy, out-of-vocab accuracy, and regained accuracy of all models on our PARA-GEO, PARA-JOB, and PARA-ATIS datasets.

| Model | GEO | | JOB | |
|---|---|---|---|---|
| | Retrained test accuracy | Accuracy change | Retrained test accuracy | Accuracy change |
| Character seq-to-seq + attn | 75.79 | −1.6 | 82 | −1.28 |
| Seq-to-seq + attn [Dong et al.] | 82.28 | −2.61 | 82.37 | −2.32 |
| Seq-to-tree + attn [Dong et al.] | 82.96 | −3.18 | 84.2 | −3.3 |
| Robust seq-to-seq + attn | 83.04 | −2.19 | 83.07 | −3.96 |
| Robust seq-to-tree + attn | 84 | −2.36 | 85.04 | −3.55 |
| Argument transfer | 85.84 | −2.34 | 84.62 | −3.97 |
| Paraphrase + arg. transfer | **87.86** | **−0.44** | **87.19** | **−1.25** |

Table 4: Table showing the percentage retrained test accuracy (10 fold average), of all models on the test set, in GEO and JOB datasets.

| ATIS | | |
|---|---|---|
| Model | Retrained test accuracy | Accuracy change |
| Character seq-to-seq + attn | 82.08 | −2.07 |
| Seq-to-seq + attn [Dong et al.] | 83.61 | −1.66 |
| Seq-to-tree + attn [Dong et al.] | 82.24 | −0.8 |
| Robust seq-to-seq + attn | 83.83 | −1.21 |
| Robust seq-to-tree + attn | 82.25 | −0.34 |
| Argument transfer | 83.34 | −1.93 |
| Paraphrase + arg. transfer | **85.35** | **0.08** |

Table 5: Table showing the percentage retrained test accuracy, of all models on the test set, in ATIS dataset.

## 5.2 Results

We now present the numerical results. We report the various accuracies of the four models presented in this paper; (1) robust sequence-to-sequence, (2) robust sequence-to-tree, (3) argument transfer model, and (4) paraphrase generation method with the argument transfer model as the base parser. We compare them with the baseline attention based sequence-to-sequence and tree models by [Dong and Lapata, 2016] as well as a character level sequence-to-sequence (with attention) model known to perform well for out-of-vocabulary words in machine translation [Luong and Manning, 2016]. In Table 2 we compare the test accuracies of all models in GEO, JOB, and ATIS datasets. We observe that our models achieve better test accuracies over prior works, sequence-to-sequence/tree models, as well as character based model. Our argument transfer model, along with robust sequence-to-tree achieves the highest accuracy in JOB, and the paraphrase generation model achieves highest accuracy in GEO dataset. Both argument transfer and paraphrase generation models achieve state-of-the-art accuracy on ATIS.

To test the resilience of the models to OOV words/phrases and their ability to learn, we evaluate their accuracies over the paraphrased datasets. We report the reference accuracy, out-of-vocab accuracy, and regained accuracy. In Table 3, we observe that our models achieve much better out-of-vocab accuracy over the baseline models. Moreover the paraphrase generation model achieves the highest out-of-vocab accuracy in all paraphrased datasets. After retraining, the robust sequence-to-tree model regain highest accuracy in the PARA-GEO, PARA-JOB datasets, while robust sequence-to-sequence regain highest accuracy on PARA-ATIS dataset.

As discussed earlier, a major disadvantage of retraining neural parsers, to learn new words and phrases, is that it can reduce its accuracy on the original test set. This can be a significant problem for many practical applications. Tables 4, 5 show the retrained test accuracy of all models, and their corresponding drop in accuracy, in GEO, JOB, and ATIS datasets. We observe that due to the novel hybrid architecture of the paraphrase generation model, it exhibits the least accuracy drop after retraining. In ATIS, the paraphrase generation model gains accuracy after retraining, achieving state-of-the art test accuracy of $85.35\%$.

**Comparison:** Our experimental evaluation indicates that in order to design a robust neural parser which can achieve high parsing accuracy, is robust to OOV words/phrases, and retain its accuracy after learning OOV words by fine-tuning, the paraphrase generation model with argument transfer model as its base parser, is the best choice. The robust sequence-to-sequence/tree parsers are better at learning new words/phrases quickly using few paraphrased examples via fine-tuning, although they suffer high accuracy drop on original test set. So does the standalone argument transfer model, although initially it has higher test accuracy. Since fine-tuning is performed using the base model parameters, this implies that optimal parameters may change significantly on learning of new OOV words. The robust sequence-to-sequence/tree models are also more prone to overfitting to OOV words and phrases upon retraining. The paraphrase generation model, which retrains only the paraphrase generator but not the base parser, enables it to maintain high accuracy even after fine-tuning. It also performs most robustly to OOV words. This increased accuracy can be attributed to the fact that a sequence-to-sequence paraphrase generator translates both OOV and rare words to high frequency words in $\mathcal{V}_T$, on which the base parser has a better performance.

## 6 Conclusion

Small vocabulary sizes of semantic parsing datasets pose a significant challenge in designing parsers suitable for real world personal agents which can encounter many out-of-vocabulary words. We study the impact of such out-of-vocabulary words on accuracy of neural semantic parsers. We propose new parser models which achieve high accuracy, as well as perform robustly to queries having out-of-vocabulary words. Our flexible and modular paraphrase generation method can also be integrated with most off-the-shelf and commercial parsers (e.g. in Alexa, Siri, Bixby), demonstrating its practical applicability. Evaluating performance of these models on more complex paraphrases, without direct lexical substitution, remains a challenge since training and test paraphrases may not have any common words/phrases, making it hard to quantify how much is learned by the model. Examining the performance of neural parsers in learning multiple words over time is also of vital interest. We would like to explore these directions in our future work.

## Acknowledgments

The authors would like to thank Li Dong for sharing the benchmark Job-queries dataset. Avik also thanks Justin Martineau for some useful discussion on experimental evaluation.

## References

[Artzi and Zettlemoyer, 2013] Yoav Artzi and Luke Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *TACL*, 1:49–62, 2013.

[Azaria *et al.*, 2016] Amos Azaria, Jayant Krishnamurthy, and Tom M Mitchell. Instructable intelligent personal agent. In *Proc. of the 30th AAAI*, pages 2681–2689, 2016.

[Bahdanau *et al.*, 2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the ICLR, San Diego, California*, 2015.

[Berant and Liang, 2014] Jonathan Berant and Percy Liang. Semantic parsing via paraphrasing. In *Proc. of ACL (1)*, pages 1415–1425, 2014.

[Bobrow, 1964] Daniel G Bobrow. Natural language input for a computer problem solving system. In *Proc., Fall Joint Comput. Conf. 25 (1964). Also available as Ph.D. Thesis, Math. Dept., MIT, Cambridge, Mass.*, 1964.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proc. of the 2014 EMNLP*, pages 1724–1734, 2014.

[Dong and Lapata, 2016] Li Dong and Mirella Lapata. Language to logical form with neural attention. In *Proc. of the 54th ACL 2016*, 2016.

[Dong *et al.*, 2017] Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. Learning to paraphrase for question answering. In *Proc. of EMNLP 2017*, pages 875–886, 2017.

[Jia and Liang, 2016] Robin Jia and Percy Liang. Data recombination for neural semantic parsing. In *Proc. of the 54th ACL*, 2016.

[Karpathy and Fei-Fei, 2017] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):664–676, 2017.

[Kiddon *et al.*, 2016] Chloé Kiddon, Luke S. Zettlemoyer, and Yejin Choi. Globally coherent text generation with neural checklist models. In *Proc. of EMNLP*, 2016.

[Krishnamurthy, 2016] Jayant Krishnamurthy. Probabilistic models for learning a semantic parser lexicon. In *Proceedings of NAACL-HLT*, pages 606–616, 2016.

[Kwiatkowski *et al.*, 2011] Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. Lexical generalization in ccg grammar induction for semantic parsing. In *Proc. of EMNLP, 2011*, pages 1512–1523. ACL, 2011.

[Luong and Manning, 2016] Minh-Thang Luong and Christopher D. Manning. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proc. of the 54th ACL 2016*, 2016.

[Luong *et al.*, 2015] Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. Addressing the rare word problem in neural machine translation. In *Proceedings of ACL 2015*, pages 11–19, 2015.

[Mikolov *et al.*, 2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, 2013.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[Prakash *et al.*, 2016] Aaditya Prakash, Sadid A. Hasan, Kathy Lee, Vivek V. Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. Neural paraphrase generation with stacked residual LSTM networks. In *Proc. of COLING 2016*, pages 2923–2934, 2016.

[Rush *et al.*, 2015] Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *Proc. of the 2015 EMNLP*, pages 379–389, 2015.

[Sutskever *et al.*, 2014] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Proc. of NIPS*, pages 3104–3112, 2014.

[Thomason *et al.*, 2015] Jesse Thomason, Shiqi Zhang, Raymond J. Mooney, and Peter Stone. Learning to interpret natural language commands through human-robot dialog. In *Proc. of IJCAI 2015,*, pages 1923–1929, 2015.

[Vinyals *et al.*, 2015a] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Proc. of NIPS*, pages 2692–2700, 2015.

[Vinyals *et al.*, 2015b] Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey E. Hinton. Grammar as a foreign language. In *Proc. of NIPS*, pages 2773–2781, 2015.

[Wong and Mooney, 2007] Yuk Wah Wong and Raymond J Mooney. Learning synchronous grammars for semantic parsing with lambda calculus. In *ACL*, volume 45, page 960, 2007.

[Zelle and Mooney, 1996] John M Zelle and Raymond J Mooney. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055, 1996.

[Zettlemoyer and Collins, 2005] Luke S Zettlemoyer and Michael Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the 21st UAI*, pages 658–666, 2005.

[Zettlemoyer and Collins, 2007] Luke S Zettlemoyer and Michael Collins. Online learning of relaxed ccg grammars for parsing to logical form. In *EMNLP-CoNLL*, pages 678–687, 2007.

[Zhao *et al.*, 2009] Shiqi Zhao, Xiang Lan, Ting Liu, and Sheng Li. Application-driven statistical paraphrase generation. In *Proc. of the 47th ACL and 4th IJCNLP-AFNLP*, pages 834–842. Association for Computational Linguistics, 2009.