Syntactic and Semantic Uniformity for Semantic Parsing and Task-Oriented Dialogue Systems

Bowen Chen
Harbin Institute of Technology
hitbwchen@gmail.com

Yusuke Miyao
The University of Tokyo
yusuke@is.s.u-tokyo.ac.jp

Abstract

This paper proposes a data representation framework for semantic parsing and task-oriented dialogue systems, aiming to achieve a uniform representation for syntactically and semantically diverse machine-readable formats. Current NLP systems heavily rely on adapting pre-trained language models to specific tasks, and this approach has been proven effective for modeling natural language texts. However, little attention has been paid to the representation of machine-readable formats, such as database queries and dialogue states. We present a method for converting original machine-readable formats of semantic parsing and task-oriented dialogue datasets into a syntactically and semantically uniform representation. We define a meta grammar for syntactically uniform representations and translate semantically equivalent functions into a uniform vocabulary. Empirical experiments on 13 datasets show that accuracy consistently improves over original formats, revealing the advantage of the proposed representation. Additionally, we show that the proposed representation allows for transfer learning across datasets.

1 Introduction

The common practice in current NLP research is to encode or decode natural language texts using large-scale pre-trained language models (Devlin et al., 2019; Liu et al., 2019; Raffel et al., 2020), which have been proven effective in modeling diverse natural language texts. In contrast with natural language, however, little attention has been paid to modeling machine-readable formats, such as database queries and dialogue states, while such formats are widely accepted in NLP tasks and developed to have specific usage like semantic parsing and task-oriented dialogue systems. Each machine-readable format is defined based on its corresponding task and dataset, but the syntactic and semantic gap among machine-readable formats is huge. This hinders the development of general methods that can be applied to diverse tasks and datasets.

To address the above problem, we propose a framework for representing machine-readable formats of diverse datasets on semantic parsing and task-oriented dialogue systems, with increased syntactic and semantic uniformity (Figure 1) to mitigate such syntactic and semantic gap among machine-readable formats. First, we define a meta grammar to build a syntactically uniform representation across various datasets. Original machine-readable formats are converted to a representation defined by this meta grammar. Further, we define mapping rules to translate semantically equivalent functions into a uniform vocabulary. This translation assigns the same form to frequently used functions such as equivalence and comparison. Combining these two methods, we can incorporate multiple data formats into a shared data representation, achieving syntactic and semantic uniformity. It should be noted that, however, the target of this work is not to create a completely unified dataset. Our goal is to achieve a high-level abstraction of diverse datasets that enables us to explore a universal model for machine-readable formats while preserving their original features.

We evaluate the proposed representation on 13 datasets of semantic parsing and task-oriented dialogue systems involving diverse machine-readable outputs, including logical forms, SQL, and dialogue states. The experiment shows that accuracy consistently improves by converting original data into the uniform representation, proving the effectiveness of the proposed framework. Further, our representation enables an easy knowledge transfer between datasets. Transfer experiments show that the model could benefit from other datasets, which improves accuracy and leads to quicker convergence in the training.

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 855–867
December 7-11, 2022 ©2022 Association for Computational Linguistics

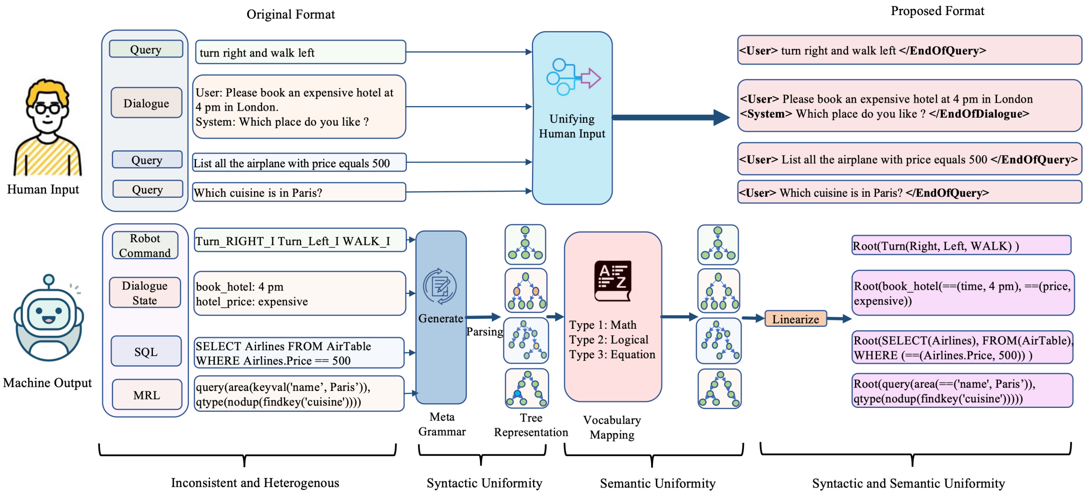

Original Format
Proposed Format
Human Input
Query
turn right and walk left
Dialogue
User: Please book an expensive hotel at 4 pm in London.
System: Which place do you like ?
Query
List all the airplane with price equals 500
Query
Which cuisine is in Paris?
Unifying Human Input
<User> turn right and walk left </EndOfQuery>
<User> Please book an expensive hotel at 4 pm in London
<System> Which place do you like ? </EndOfDialogue>
<User> List all the airplane with price equals 500 </EndOfQuery>
<User> Which cuisine is in Paris? </EndOfQuery>
Machine Output
Robot Command
Turn_RIGHT_I Turn_Left_I WALK_I
Dialogue State
book_hotel: 4 pm
hotel_price: expensive
SQL
SELECT Airlines FROM AirTable
WHERE Airlines.Price == 500
MRL
query(area(keyval('name', Paris')),
qtype(nodup(findkey('cuisine'))))
Generate
Parsing
Meta Grammar
Tree Representation
Type 1: Math
Type 2: Logical
Type 3: Equation
Vocabulary Mapping
Linearize
Root(Turn(Right, Left, WALK))
Root(book_hotel(==(time, 4 pm), ==(price, expensive))
Root(SELECT(Airlines), FROM(AirTable), WHERE (==(Airlines.Price, 500)))
Root(query(area(==('name', Paris')),
qtype(nodup(findkey('cuisine')))))
Inconsistent and Heterogenous
Syntactic Uniformity
Semantic Uniformity
Syntactic and Semantic Uniformity

Figure 1: Proposed Data Representation Framework.

2 Related Work

2.1 Semantic Parsing and Task-Oriented Dialogue Systems

Semantic parsing (Kamath and Das, 2018) is a task to transform a user's utterance into a machine-readable format. An illustrating application is database QA, in which a system translates a natural language utterance into a query language for a given database. A variety of datasets have been proposed, where the representation format of database queries is diverse; examples include SQL, Prolog, and MRL (Machine Readable Language) (Chamberlin and Boyce, 1974; Bratko, 2012; Lawrence and Riezler, 2016a). Another application of semantic parsing is to convert a natural language text into some form of meaning representation. For example, SCAN (Lake and Baroni, 2018) is a dataset of robot command utterances paired with their meaning representations. Semantic parsing datasets are developed under the shared goal, i.e., convert natural language texts into a machine-readable format, while data representation specifics are varied.

Task-oriented dialogue systems interact with a user in natural language to achieve a specific goal like buying tickets or booking a hotel. A system tracks the user's intent from the dialogue, queries a database, and generates a response according to the result. Previous approaches use belief state representations, a.k.a. dialogue states, to represent the user's intent as slot-value pairs. Most datasets on task-oriented dialogue systems adopt this representation, including MultiWOZ (Budzianowski et al., 2018), DSTC2 (Henderson et al., 2014), ATIS (Hemphill et al., 1990), and M2M (Shah et al., 2018). A closely related task called conversational semantic parsing proposed recently represents user intent as a program. Cheng et al. (2020) proposes TreeDST, in which user intent is represented as a hierarchical format program. SMCalFlow (Andreas et al., 2020) further develops this idea, where the representation involves a mechanism to track a user's previous information and retrieve it when necessary to handle dialogue-specific phenomena such as revision. All the datasets listed above use machine-readable formats to track user intent, while data representations diverge considerably.

The focus of the present research is to minimize the diversity of machine-readable formats of semantic parsing and dialogue systems discussed above. It should be noted that, the present paper does not work on complete dialogue systems involving response generation. However, previous research (Rastogi et al., 2017; Mrkšić et al., 2017; Budzianowski et al., 2018) proved that the performance of dialogue state tracking is essential for the entire system performance because the model is likely to generate incorrect responses when the dialogue state is wrongly predicted.

2.2 Cross-Dataset Evaluation and Cross-Task Model

Another area related to the present work is cross-dataset evaluation, which aims to evaluate a model not only on in-domain test set accuracy but also on generalization ability to out-domain data. Some work focuses on the image classification area (Yang et al., 2019; Hoffman et al., 2018). In the NLP filed, Chen et al. (2020) proposes a cross-evaluation method in the summarization task to test whether the model can still summarize new data. Nejadgholi and Kiritchenko (2020) proposes a cross-evaluation method in abusive language detection as most abusive languages are sampled from similar phrases, thus leading to a weak generalization ability. However, the above work supposes that the data representation format is the same across different datasets, which we cannot assume for semantic parsing and task-oriented dialogue systems. Our research further extends the idea of cross-dataset evaluation by converting different formats from diverse datasets into a uniform representation.

Another series of research explores methods for using pre-trained language models or text-to-text models, such as T5 (Raffel et al., 2020), for diverse tasks. Xie et al. (2022) proposes UnifiedSKG, which extends T5 to encode not only text input but also structured representations like a database and machine-readable formats. The basic idea is to convert such structured representations into natural language expressions so that T5 can process them directly. Their goal is to build a unified model for diverse NLP tasks, including semantic parsing and dialogue systems. However, they do not intend to solve the problem of the diversity of representation formats of datasets. Another issue in this framework is that target representations are inherently limited to shallow structures. This is because encoding complex structured representations in natural language are problematic, such as a program with a deep tree structure like SMCalFlow (Andreas et al., 2020). Their method and the present research focus on different aspects of the diversity of tasks and datasets. These ideas can possibly be integrated to build a unified model for processing diverse datasets and machine-readable formats.

3 Uniform Representation

Figure 1 shows the overview of our proposal. Datasets of semantic parsing and task-oriented dialogue systems consist of pairs of input and output instances. We call a natural language query in semantic parsing and a dialogue in task-oriented dialogue systems the *human input*. A machine-readable output, such as database queries and dialogue states, is called the *machine output*. As shown in this figure, machine outputs are given in heterogeneous representations in different datasets. Our goal is to convert original representations of machine outputs into a syntactically and semantically uniform representation.

3.1 Syntactic Uniformity

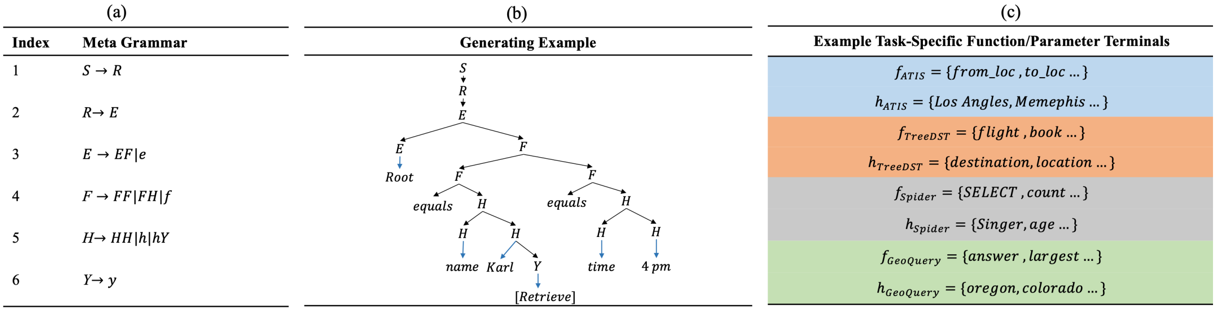
In order to achieve syntactic uniformity, we define a meta grammar as a higher-level abstraction of the grammar used for each dataset. The meta grammar provides the basic syntactic structure of the proposed representation. A grammar for each specific dataset will be derived by extending the meta grammar as we describe below. The meta grammar $G = (V, T, P, S)$ is defined as:

- V is a set of variables $\{R, E, F, H, Y\}$, which represent the root variable, head token variable, function variable, parameter variable and attribute variable, respectively.
- T is a set of terminal symbols e, f, h, y, which represent the head-token, function, parameter, and attribute, respectively.
- P is the set of production rules as given in Figure 2 (a).
- S is a start variable.

Starting from generating a root variable R (Rule 1), the grammar generates a head token E and a function F (Rule 2 and 3). A function can be generated multiple times and can accompany by a parameter H (Rule 4). A parameter can also be generated multiple times and can accompany an attribute Y (Rule 5). E, F, H, and Y generate terminal symbols e, f, h, and y, respectively (Rule 3, 4, 5, 6). Figure 2 (b) shows an example derivation tree, where terminal symbols are replaced with task-specific symbols as explained below.

A grammar for a specific dataset is obtained by replacing terminal symbols, i.e., e, f, h, and y, with tokens used in the dataset. Figure 2 (c) shows examples of task-specific terminal symbols. For example, f_{ATIS}, which corresponds to the terminal symbol f for the ATIS dataset, is defined as a set of all function names in this dataset.

The grammar for each dataset generates a sequence of terminal symbols, which corresponds to a token sequence of each dataset. Token sequences are finally converted into a tree structure, in a similar way to converting a phrase structure tree into a dependency tree. We regard elements of e (head token) and f (function) as head and other tokens as dependents. That is, tokens of e and f are placed in internal nodes, and their sibling nodes become their subtrees. An attribute is attached to a corresponding token. The example tree of Figure 2 (b) is converted into the following format:


(a)

Index	Meta Grammar
1	$S \rightarrow R$
2	$R \rightarrow E$
3	$E \rightarrow EF \mid e$
4	$F \rightarrow FF \mid FH \mid f$
5	$H \rightarrow HH \mid h \mid hY$
6	$Y \rightarrow y$

(b)

Generating Example

(c)

Example Task-Specific Function/Parameter Terminals
$f_{ATIS} = \{from\_loc , to\_loc \ldots\}$
$h_{ATIS} = \{Los\ Angles, Memephis \ldots\}$
$f_{TreeDST} = \{flight , book \ldots\}$
$h_{TreeDST} = \{destination, location \ldots\}$
$f_{Spider} = \{SELECT , count \ldots\}$
$h_{Spider} = \{Singer, age \ldots\}$
$f_{GeoQuery} = \{answer , largest \ldots\}$
$h_{GeoQuery} = \{oregon, colorado \ldots\}$

Figure 2: The meta grammar (a), an example derivation (b), and task-specific terminals (c).

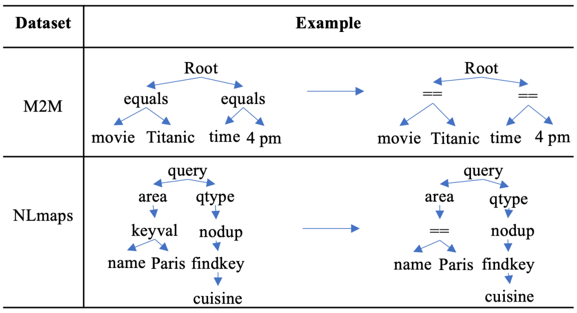


Figure 3: Examples of Vocabulary Mapping

```
Root(equals(name, Karl[Retrieve]),
     equals(time, 4pm))
```

3.2 Semantic Uniformity

After generating the machine output following the meta grammar, we have a syntactically uniform representation for different datasets. However, achieving syntactic uniformity does not lead to semantic uniformity, which means semantically equivalent tokens are denoted with the same representation. For example, the function to denote equality is represented as `keyval` in NLmaps but as `equals` in TreeDST.

To achieve semantic uniformity, we focus on three types of functions, namely, mathematical, logical, and equation operators, that are frequently used in diverse datasets. We define uniform vocabularies, t, l, and e, for these functions as below.

$$t = \{+, -, *, /\}$$
$$l = \{and, or, negate\}$$
$$e = \{==, >, <, >=, <=, ! =\}$$

These functions have the same meaning across various datasets but are denoted differently. We, therefore, convert the representation by mapping function names in the original form into the uniform vocabulary defined above. If a dataset contains a mathematical, logical, or equation function, we replace the function name with its corresponding symbol from the uniform vocabulary set, i.e., t, l, or e. Examples in Figure 3 show that `equals` in M2M and `keyval` in NLmaps are replaced by the symbol == in the uniform symbol set e_m.

3.3 Multi-Turn Interactions

People often refer to or revise previous information in multi-turn interactions. Therefore, it is necessary for our format to have a mechanism to represent such structures. We use the attribute symbol y defined in Section 3.1 to achieve this. Consider the following example from SMCalFlow:

```
Root(create_event(Location,
                  Pax Square))
Root(do(revise<(Event)>(
            Location,
            new<Event>(
                Location,
                Red Plaza))))
```

In this example, the user first creates an event in `Pax Square` but revises it to `Red Plaza` in the following conversation. To represent this, we add a special attribute at the leaf node, which is `[Retrieve]`. This special token indicates that it might be retrieved in the following turn. Additionally, we put the `[StartRetrieve]` symbol at functions that retrieves previous information like `revise`. The above program is converted as:

```
Root(create_event(Location,
                  Pax Square
                  [Retrieve]))
Root(do(revise<(Event)>
            [StartRetrieve](
            Location,
            new<Event>(
                Location,
                Red Plaza))))
```

By adding the special symbols, our format can navigate to the part that triggers the retrieval of previous information. This conversion is applied only when the original dataset provides such information.

3.4 Uniform Human Input

We also aim to represent different human inputs in a uniform way since the human input also has some diversity depending on task definitions. For example, in task-oriented dialogue datasets, the input contains multi-turn interactions between humans and the dialogue system. If the interaction between the human and the system is described without structural information, the model cannot understand which utterance is from the user or the system. The following symbols are introduced to give necessary structures to human input:

<User> means that the utterance after it comes from a user.

<System> means that the utterance after it comes from the response of a system.

</EndOfDialogue> indicates the end of a multi-turn dialogue between a human and a system.

</EndOfQuery> indicates the end of a user query.

3.5 Task-Specific Processing

For datasets with tree-structured machine output, like GeoQuery and Spider, we directly parse them and convert them to follow our grammar rules. The following sections introduce details of processing steps for specific datasets.

3.5.1 Processing SCAN

An example machine output of SCAN is:

```
TURN_RIGHT, WALK, TURN_RIGHT,
WALK, TURN_RIGHT, RUN, JUMP
```

The machine output of SCAN is formed by the combination of `TURN_X`, where `X` is a token denoting a direction like `RIGHT` and `LEFT`, and independent actions like `WALK` and `RUN`. Therefore, `TURN` can be considered as a function, while `X` and actions like `WALK` and `RUN` as a parameter. Thus, we split the original machine output by the `TURN` function and make other tokens as a parameter of the preceding `TURN`. The above example is converted into:

```
Root(Turn(Right, Walk),
     Turn(Right, Walk),
     Turn(Right, Run, Jump))
```

3.5.2 Processing Belief State Representations

An example of a belief state representation of task-oriented dialogue systems is like:

```
inform hotel-area: east;
inform hotel-parking: yes;
inform hotel-stars: 4;
```

In this example, `inform` can be considered as a function of the dialogue system. The expression `hotel-area: east` can be seen as the information that the dialogue state tracking module transmits to the following dialogue module. Here, we cannot see `hotel-area: east` as a nested function such that the `hotel-area` is a function and the `east` is a parameter. This expression is regarded that `east` is a value assigned to the variable `hotel-area`. Therefore, we give a representation like `hotel-area == east`.

3.5.3 Disambiguation in TreeDST

Another phenomenon to note is an ambiguity issue in TreeDST. In this dataset, `equals` is treated differently in different contexts. An original example in TreeDST is given below:

```
Root(find(
     object(
      equals(
       equals(dateTimeRange,
        equals(startDateTime,
         equals(date,
          equals(Thursday,
                 dayOfWeek)))))))
```

In this example, the `equals` function not only means equality but also works as the `and` logical operator, which requires the object to meet multiple conditions. We disambiguate the meaning of `equals` in the following manner. The parameters of an equality operator node cannot contain equality operators. Otherwise, we rewrite the equality operator to logical operator `and`.

3.5.4 Processing SMCalFlow

We process the Dataflow structure of SMCalFlow to simplify expressions that cannot be executed under available resources. The main motivation is that the Dataflow structure highly relies on the target database and the Dataflow executor, which are not publicly released. This means that many expressions of the Dataflow structure do not have actual semantics. To remove such expressions, we apply the following process.

1. Remove APIs that require querying a database or executing a program, like `QueryEventResponse.results`. This is because external APIs in the Dataflow structure require triggering an executor or the

database, but the executor and the database are unavailable.

2. Simplify the Dataflow structure by adding a syntax sugar for a `constraint`. A constraint restricts the input of a function, to trigger an exception when a function receives an undesired input. We keep the constraint mechanism but remove functions that do not have actual semantics. For example, an example of the original constraint is like: `refer(^(Event) EmptyStructConstraint)`. This structure specifies that the return value of the external API `EmptyStructConstraint` has to be the `Event` type. By converting this structure, we have `refer<(Event)>` to represent the input of `refer` is `Event`.

3.6 Back Transformation

Following the above process, we can transform each dataset and represent them in syntactically and semantically uniform data representations. This conversion process does not lose any information for most datasets. The only case is SMCalFlow, in which we simplified its expression. Therefore, we can easily transform them back to their original format of the machine output to enable evaluation in the original datasets.

4 Experiment

This section describes experiment results to evaluate the effectiveness of the proposed uniform representation. It should be noted that we do not propose any new models in this paper, while most SOTA methods for each dataset are based on specifically designed models or external knowledge. We use simple baseline models while changing the representation of machine outputs of each dataset. We describe datasets and evaluation metrics we use for the experiments, followed by results on in-domain experiments and transfer experiments.

4.1 Dataset Description

For semantic parsing datasets, we select GeoQuery (Zelle and Mooney, 1996), Spider (Yu et al., 2018), and NLmaps (Lawrence and Riezler, 2016b) datasets, which include three types of query language: Prolog, SQL, and MRL. We also select compositional parsing datasets, including SCAN-Si and SCAN-LEN, where the latter requires models to learn from short robot commands and generalize to longer commands. We also choose CoSQL (Yu et al., 2019a) and SParC (Yu et al., 2019b) datasets, in which the user queries the database through multiple turns. The CoSQL contains the dialogue between the user and the system, whereas the SParC only contains the utterance from the user.

For the task-oriented dialogue datasets, we select ATIS, M2M, DSTC2, and MultiWOZ datasets (Hemphill et al., 1990; Shah et al., 2018; Henderson et al., 2014; Budzianowski et al., 2018). ATIS consists of conversations in an airplane booking system. M2M includes conversations in the restaurant and movie domains. DSTC2 mainly targets the restaurant domain. The MultiWOZ dataset covers several domains and has more interactions between humans and systems, and we chose its newest release, e.g., the MultiWOZ2.2. Machine outputs of these datasets use belief state representations. We also choose TreeDST and SMCalFlow to represent a different paradigm to express user intent in task-oriented dialogue systems, in which the user intents are a program with a hierarchical structure.

4.2 Data Statistics

Table 1 shows the statistics of the datasets we use for evaluation, including train, validation, and test size, and the vocabulary size of the human input (H) and machine output (M). We use the OpenNMT tool to process these datasets (Klein et al., 2017). NA indicates that the test set is unavailable for testing, and we report evaluation results on the validation set. The table also shows the task type of each dataset.

From the table, we can see the vocabulary size of machine output is less than the human input for SCAN, ATIS, GeoQuery, and TreeDST datasets, which is different from machine translation, where the vocabulary sizes are similar for the source language and target language. SMCalFlow, SParc, and CoSQL have a close or a larger target vocabulary size, and this is due to the data creation process containing many entity words like `JetBlue Airway` that only appear once in the validation set.

Since the proposed format transforms the dataset into a tree representation, we also calculated the average tree depth in all datasets that we used. Notice that the tree depth only means the depth of nested trees and does not mean the linearized tree has a short length as a root could have multiple and parallel sub-trees.

Dataset	Train Size	Validation Size	Test Size	Vocab Size (H/M)	Task Type	Tree Depth
ATIS	4,959	993	891	886 / 616	DST	3.00
MultiWOZ2.2	56,668	7,374	7,368	15,459 / 988	DST	2.97
M2M	1,500	4,69	1,039	482 / 100	DST	2.93
DSTC2	11,677	3,934	9,890	390 / 118	DST	3.00
TreeDST	19,808	3,733	3,739	6,679 / 1,367	CSP	7.09
SMCalFlow	121,200	13,499	NA	12,319 / 13,218	CSP	8.13
Spider	7,000	1,034	NA	3,401 / 4,827	SQL Parsing	4.05
SParC	12,059	1,625	NA	5,521 / 5,699	SQL Parsing	4.06
CoSQL	9,502	1,300	NA	6,227 / 6,953	SQL Parsing	3.92
SCAN-Si	13,383	3,345	4182	14 / 7	Compositional Parsing	2.87
SCAN-Len	16,690	3,345	4,920	14 / 7	Compositional Parsing	2.99
NLmaps	16,172	1,843	10,593	6,059 / 5,266	MRL Parsing	5.04
GeoQuery	540	60	280	246 / 163	Prolog Parsing	5.84

Table 1: Statistics of the datasets. DST indicates dialogue state tracking and CSP indicates conversational semantic parsing.

4.3 Models and Evaluation Metrics

The models we use for the evaluation are:

Bi-LSTM The encoder-decoder model with bidirectional LSTM (Hochreiter and Schmidhuber, 1997). We use the copy mechanism (Gu et al., 2016) and 300-dimensional GloVe (Pennington et al., 2014) embeddings.

TreeDec LSTM This is a variant of LSTM, in which the model can begin a hierarchical decoding process by adding a <N> symbol to indicate the decoding of a subtree (Dong and Lapata, 2016).

BERT2Seq We use the BERT-base model as the encoder and an LSTM plus the copy mechanism as the decoder.

The evaluation metrics used in this paper are:

Sentence Level Exact Match This metric measures the ratio of the outputs that are exactly the same as the gold truth.

Joint Goal Accuracy This metric is used in dialogue state tracking tasks. It compares a predicted belief state representation to the ground truth. The prediction is considered correct when the predicted values exactly match the ground truth.

Set Match This is the metric for SParC, CoSQL, and Spider, evaluating an SQL query as a set to leave space for order-insensitive functions.

Denotation Accuracy This metric executes a program and checks the return value with the ground truth.

Word-Level EM with/without Refer The metric of SMCalFlow, a word-level exact match with/without considering the refer relation.

4.4 Main Results

Table 2 shows the results of the experiment. Note that the results for SMCalFlow are not comparable with the previous method since the original data is simplified as described in Section 3.5.4. We can see the following observations from the results:

1. The proposed data format improves the performance of all models in all tasks, showing the effectiveness of the proposed uniform representation framework. This performance gain using format conversion is almost effortless comparing to designing a new model or increasing a data size.

2. Our representation with simple models outperforms SOTA on TreeDST, DSTC2, and NLmaps.

3. TreeDec LSTM improves the results of the Bi-LSTM model showing that a tree-structured decoding method is effective for the proposed framework.

4. The results on SCAN-Len show that the compositional task can benefit from our representation, while it does not achieve complete generalization.

5. We can see some improvements on Spider, SParC, and CoSQL, while the models cannot solve these tasks. Incorporating external knowledge like the target database is necessary to achieve sufficient accuracy on these datasets.

6. The results reveal that the performance does not have obvious relationship with the average

Dataset	Metric	Bi-LSTM		TreeDec		BERT2Seq		SOTA	E/S
		Ori	Tree	Ori	Tree	Ori	Tree	–	
ATIS	Joint Acc	84.3	85.6	–	86.9	84.1	87.2	**88.6**	✗
MultiWOZ2.2		44.9	45.5	–	46.2	45.3	48.1	**54.4**	✓
M2M		74.1	74.8	–	75.4	84.1	87.8	**90.9**	✓
DSTC2		79.6	81.1	–	81.3	84.6	**86.8**	85.0	✓
SMCalFlow	EM with Refer	71.2	73.0	–	74.1	72.8	**74.8**	73.8	✗
	EM without Refer	72.0	74.4	–	76.4	74.3	**77.1**	75.3	
Spider	Set EM	1.4	1.6	–	1.7	1.5	2.1	**75.1**	✓
SParC		1.1	1.3	–	1.4	1.4	2.0	**48.5**	✓
CoSQL		1.0	1.2	–	1.3	1.2	1.4	**24.6**	✓
TreeDST	Sent-Level EM	57.7	60.3	–	62.3	60.8	**62.8**	62.2	✗
SCAN-Si		98.3	99.8	–	100	100	**100**	**100**	✓
SCAN-Len		13.3	16.2	–	16.3	3.1	6.3	100	✓
NLmaps		65.1	66.3	–	66.9	68.2	**70.7**	69.3	✗
GeoQuery	Denotation Acc	68.5	70.3	–	72.1	81.1	84.3	**86.1**	✓

Table 2: Main Results. Ori means original format. Tree means the proposed representation. E/S means whether the SOTA model uses extra information or a specifically designed algorithm. SOTA for GeoQuery and SCAN are from Herzig and Berant (2021). SOTA for NLmaps is from Damonte et al. (2019). SOTA for TreeDST and SMCalFlow are from Cheng et al. (2020) and Andreas et al. (2020). SOTA for Spider is from Scholak et al. (2021). SOTA for SParC and CoSQL is from Xiao et al. (2022). SOTA for MultiWOZ2.2, M2M and DSTC2 are from Feng et al. (2021). SOTA for ATIS is from Chao and Lane (2019). Bold texts indicate the best results.

Train	Target	BERT2Seq	
		P(%)	C(%)
DSTC2	DSTC2	86.6	86.8
MW+DSTC2	DSTC2	87.7	60.8
MW+M2M+DSTC2	DSTC2	88.3	53.7
MW+M2M+ATIS+DSTC2	DSTC2	**89.8**	**42.6**
ATIS	ATIS	85.1	87.6
MW+ATIS	ATIS	87.1	59.6
MW+M2M+ATIS	ATIS	87.8	47.5
MW+M2M+DSTC2+ATIS	ATIS	**88.7**	**40.2**
M2M	M2M	87.5	84.4
MW+M2M	M2M	87.9	68.7
MW+M2M+DSTC2	M2M	88.3	44.3
MW+M2M+DSTC2+ATIS	M2M	**88.6**	**18.9**
NLmaps	NLmaps	70.2	87.3
GeoQuery+NLmaps	NLmaps	**70.6**	**69.2**

Table 3: Transfer Experiment. MW is MultiWOZ2.2.

tree depth. This indicates that each dataset has its inherent difficulty based on its task definition, and those difficulties are not changed by the format conversion.

4.5 Transfer Experiments

Machine outputs of all the datasets in our representation have syntactic and semantic uniformity. Therefore, we expect models learned in the uniform representation can be transferred to other datasets.

In this experiment, we select MultiWOZ2.2, DSTC2, M2M, and ATIS as the datasets for task-oriented dialogues. We also select NLmaps and GeoQuery as the datasets for semantic parsing. We train the model with and without external datasets.

We evaluate the task performance and convergence speed when transferring to a new dataset, represented as *P* and *C*. *P* means the task performance of the model, and *C* means the convergence step divided by the scheduled training step (60, 000). The early stopping with a tolerance step of eight is used to find the convergence step. We keep the model parameter the same in all experiments; therefore, the accuracy numbers of the single dataset experiment are slightly different from the results reported in Table 2. From the results in Table 3, we can see:

1. Task performances consistently improve by transfer learning. All transfer results outperform the previous results in Table 2.

2. The transfer experiments of the task-oriented dialogue datasets reveal that MultiWOZ2.2 contributes more than other datasets. This is possibly because MultiWOZ2.2 contains multi-domain dialogues and has higher average interaction turns. This can make MultiWOZ2.2 more informative than other datasets.

3. The results also reveal quicker convergence in all configurations. This shows that the proposed representation makes it easier for the

Error Type	Count	Percentage
Redundant Function	27	13.5%
Wrong Parameter	56	28.0%
Incomplete Generation	33	16.5%
Repetitive Generation	24	12.0%
– Parameter	14	7.0%
– Function	10	5.0%
Grammatical Error	43	21.5%
– Correlated Statement	22	11.0%
– Missing Parameter	15	7.5%
– Unmatch Parenthesis	6	3.0%
Unseen Token	17	8.5%

Table 4: Error Analysis on ATIS

model to transfer the knowledge from other datasets to the target dataset.

4.6 Error Analysis

We manually collected 200 wrong examples from the ATIS dataset's prediction to analyze the remaining issues. Table 4 shows the results. The *Redundant Function* means the model generates an unwanted extra function. The *Wrong Parameter* means the model produces a grammatically correct prediction but with a wrong parameter. The *Incomplete Generation* means the model's generation lacks a function or a parameter. The *Repetitive Generation* means the model repeatedly outputs a function or a parameter. The *Grammatical Error* means the model's prediction has grammar errors, in which the *Correlated Statement* errors mean the model did not understand some correlated functions. The *Unseen Token* error means the model cannot generate the out-of-vocabulary symbol.

From the results, we can see that the major error type is the wrong parameter error, which implies that machine outputs are grammatically correct but inconsistent with human inputs. The incomplete generation error is possibly due to test instances that are deviated from the majority of training data, and the model terminates without a complete generation of the machine output. Another important error type is the correlated statement error. For example, the function $from\_loc$ of ATIS usually is followed by to_loc, whereas the model cannot understand such correlated statement and only generates one of them. Other error types are common errors in sequence-to-sequence frameworks, like the model cannot generate out-of-vocabulary words or the model making repetitive generation of a certain token.

These errors indicate that the current model is limited regarding two aspects. First, the language understanding part of the model is insufficient to correctly understand the human input. This causes the model produces grammatically correct outputs but with wrong parameters. Advanced encoding models like T5 (Raffel et al., 2020) and GPT-2 (Radford et al., 2019) are expected to reduce such errors. Second, the model cannot effectively learn a dataset-specific grammar from the training data. This is mainly due to two reasons. First, the model does not have any prior or external knowledge about the grammar of each dataset. Our uniform representation partially solves this problem, although it still requires sufficient amount of train data to learn the task-specific grammar. Additionally, the model does not include any mechanism to produce only grammatically correct outputs. However, the proposed method brings structural information by converting the original format to a tree representation, which naturally makes the model learn to make structural predictions through training. This leads to the lowest percentage of *Unmatch Parenthesis* among all error types, showing the effectiveness of the proposed method.

5 Conclusion

This research presents a syntactically and semantically uniform data representation framework for semantic parsing and task-oriented dialogue systems. Using our framework, we can incorporate both human input and machine output in 13 datasets into a both syntactically and semantically uniform representation. The experiments show promising improvements in multiple datasets compared to original data representations and even outperform several SOTA performances showing the benefit brought by the proposed format. Additionally, our representation framework enables a convenient knowledge transfer. Transfer experiment results show that the knowledge learned from other tasks can be easily transferred to the target task, improving the results on the target task and the convergence speed.

6 Limitations

This paper is focused on a syntactically and semantically uniform representation that can be applied to diverse datasets of semantic parsing and task-oriented dialogue systems but does not propose any new models or architectures for these tasks. We adopted simple sequence-to-sequence models in the experiments rather than state-of-the-art methods for each task because our purpose is to evaluate data representations across diverse datasets rather than evaluating models. However, the experiments showed that the models could not solve several datasets, including SCAN-Len, Spider, SParC, and CoSQL, indicating that we need task-specific algorithms or encoding of external information like target databases. This means dataset/task-specific methods are indispensable even with our proposed representation. The integration of our representation and dataset/task-specific methods is possible while it is left for future work. Additionally, the proposed format is not proven optimal, while the experiment showed improvements on diverse datasets. Further extensions of the format, e.g. using a graph format rather than a tree, can be explored in the future.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number JP19H05692.

References

Jacob Andreas, John Bufe, David Burkett, Charles Chen, Josh Clausman, Jean Crawford, Kate Crim, Jordan DeLoach, Leah Dorner, Jason Eisner, Hao Fang, Alan Guo, David Hall, Kristin Hayes, Kellie Hill, Diana Ho, Wendy Iwaszuk, Smriti Jha, Dan Klein, Jayant Krishnamurthy, Theo Lanman, Percy Liang, Christopher H. Lin, Ilya Lintsbakh, Andy McGovern, Aleksandr Nisnevich, Adam Pauls, Dmitrij Petters, Brent Read, Dan Roth, Subhro Roy, Jesse Rusak, Beth Short, Div Slomin, Ben Snyder, Stephon Striplin, Yu Su, Zachary Tellman, Sam Thomson, Andrei Vorobev, Izabela Witoszko, Jason Wolfe, Abby Wray, Yuchen Zhang, and Alexander Zotov. 2020. Task-oriented dialogue as dataflow synthesis. *Transactions of the Association for Computational Linguistics*, 8:556–571.

Ivan Bratko. 2012. Prolog programming for artificial intelligence, 4th edition.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

Donald D. Chamberlin and Raymond F. Boyce. 1974. Sequel: A structured english query language. In *SIGFIDET '74*.

Guan-Lin Chao and Ian Lane. 2019. BERT-DST: Scalable End-to-End Dialogue State Tracking with Bidirectional Encoder Representations from Transformer.

Yiran Chen, Pengfei Liu, Ming Zhong, Zi-Yi Dou, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2020. CDEvalSumm: An empirical study of cross-dataset evaluation for neural summarization systems. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3679–3691, Online. Association for Computational Linguistics.

Jianpeng Cheng, Devang Agrawal, Héctor Martínez Alonso, Shruti Bhargava, Joris Driesen, Federico Flego, Dain Kaplan, Dimitri Kartsaklis, Lin Li, Dhivya Piraviperumal, Jason D. Williams, Hong Yu, Diarmuid Ó Séaghdha, and Anders Johannsen. 2020. Conversational semantic parsing for dialog state tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8107–8117, Online. Association for Computational Linguistics.

Marco Damonte, Rahul Goel, and Tagyoung Chung. 2019. Practical semantic parsing for spoken language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Industry Papers)*, pages 16–23, Minneapolis, Minnesota. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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.

Yue Feng, Yang Wang, and Hang Li. 2021. A sequence-to-sequence approach to dialogue state tracking. 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 1714–1725, Online. Association for Computational Linguistics.

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

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

Matthew Henderson, Blaise Thomson, and Jason D. Williams. 2014. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272, Philadelphia, PA, U.S.A. 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*, 9:1735–80.

Steven Hoffman, Renu Sharma, and Aran Ross. 2018. Convolutional neural networks for iris presentation attack detection: Toward cross-dataset and cross-sensor generalization. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1701–17018.

Aishwarya Kamath and Rajarshi Das. 2018. A survey on semantic parsing.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Brenden M. Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *ICML*.

Carolin Lawrence and Stefan Riezler. 2016a. NLmaps: A natural language interface to query OpenStreetMap.

Carolin Lawrence and Stefan Riezler. 2016b. NLmaps: A natural language interface to query OpenStreetMap. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 6–10, Osaka, Japan. The COLING 2016 Organizing Committee.

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.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1777–1788, Vancouver, Canada. Association for Computational Linguistics.

Isar Nejadgholi and Svetlana Kiritchenko. 2020. On cross-dataset generalization in automatic detection of online abuse. In *Proceedings of the Fourth Workshop on Online Abuse and Harms*, pages 173–183, Online. Association for Computational Linguistics.

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

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Abhinav Rastogi, Dilek Hakkani-Tur, and Larry Heck. 2017. Scalable multi-domain dialogue state tracking. In *Proceedings of IEEE ASRU*.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Pararth Shah, Dilek Hakkani-Tür, Gokhan Tür, Abhinav Rastogi, Ankur Bapna, Neha Nayak, and Larry Heck. 2018. Building a conversational agent overnight with dialogue self-play. *arXiv preprint arXiv:1801.04871*.

Dongling Xiao, Linzheng Chai, Qian-Wen Zhang, Zhao Yan, Zhoujun Li, and Yunbo Cao. 2022. CQR-SQL: Conversational Question Reformulation Enhanced Context-Dependent Text-to-SQL Parsers.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I. Wang, Victor Zhong, Bailin Wang, Chengzu Li, Connor Boyle, Ansong Ni, Ziyu Yao, Dragomir Radev, Caiming Xiong, Lingpeng Kong, Rui Zhang, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2022. UnifiedSKG:

Unifying and Multi-Tasking Structured Knowledge Grounding with Text-to-Text Language Models.

Dan Yang, Veli-Tapani Peltoketo, and Joni-Kristian Kamarainen. 2019. CNN-Based Cross-Dataset No-Reference Image Quality Assessment. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*.

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

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

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

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI, Vol. 2*.

A Appendix

A.1 Main Experiment Setup

For the main experiment results, the Bi-LSTM model and the TreeDec Bi-LSTM model are based on the encoder-decoder architecture with either 3 or 4 layers with the hidden size of 368. The dropout ratio is 0.5. We train the model using AdamW (Loshchilov and Hutter, 2017) with the learning rate from $\{5\times10^{-4}, 1\times10^{-3}, 5\times10^{-3}\}$, the batch size from $\{32, 64\}$ and the total training steps from $\{2\times10^{4}, 3\times10^{4}, 5\times10^{4}, 6\times10^{4}\}$. We evaluate the model at every 500 steps to find the best model.

The BERT2Seq model consists of BERT-base as the encoder, and LSTM with 3 or 4 layers as the decoder. The hidden size of LSTM is 386. The dropout ratio is 0.5. We train the model using AdamW with the learning rate from $\{1\times10^{-5}, 3\times10^{-5}, 5\times10^{-5}\}$, the batch size from $\{32, 64\}$, and the total training steps from $\{3\times10^{4}, 5\times10^{4}, 6\times10^{4}\}$. We also use the linear learning rate schedule with a warm-up ratio of 0.1.

For the selection of best hyper parameters, we try combinations of hyper parameters listed above and evaluated the performance of each model on the validation set.

The experiment is run on NVIDIA A100 GPU with 40GB memory. The running time largely depends on the scheduled training steps based on the size of different datasets. Depending on the configuration (model, training steps, batch size, evaluation steps etc.), the training time spans from 8 hours to 48 hours.

A.2 Transfer Experiment Setup

Train	Target	Bi-LSTM	
		P(%)	C(%)
DST2	DST2	81.3	85.3
MW+DST2	DST2	82.4	63.2
MW+M2M+DST2	DST2	83.8	50.3
MW+M2M+ATIS+DST2	DST2	**85.3**	**42.9**
ATIS	ATIS	85.1	87.6
MW+ATIS	ATIS	86.1	57.3
MW+M2M+ATIS	ATIS	86.5	48.95
MW+M2M+DST2+ATIS	ATIS	**87.6**	**46.2**
M2M	M2M	74.8	86.4
MW+M2M	M2M	75.6	50.2
MW+M2M+DST2	M2M	75.8	46.5
MW+M2M+DST2+ATIS	M2M	**76.3**	**40.2**
NLmaps	NLmaps	66.3	86.2
GeoQuery+NLmaps	NLmaps	**67.4**	**76.7**

Table 5: Transfer Experiment of Bi-LSTM

The BERT2Seq model in the transfer experiment consists of BERT-base as the encoder and 3-layer LSTM with the hidden size of 384 as the decoder. The Bi-LSTM model consists of 3-layer LSTMs with the hidden size of 384 as the encoder and the decoder. For transferring to a new dataset, we update the vocabulary, keep the trained embeddings from the old vocabulary, and initialize embeddings of new tokens, so that embeddings learned in the previous model are reused. The result of the transfer experiment for the Bi-LSTM model is shown in Table 5. The result shows the same trend as BERT2Seq in the performance and the convergence steps.