# Exploring the Secrets Behind the Learning Difficulty of Meaning Representations for Semantic Parsing

**Zhenwen Li**[◇], **Jiaqi Guo**[§*], **Qian Liu**[†*], **Jian-Guang Lou**[♡], **Tao Xie**[◇*]
[◇]Peking University, Beijing, China; [§]Xi'an Jiaotong University, Xi'an, China
[†]Sea AI Lab, Singapore; [♡]Microsoft Research Asia, Beijing, China
lizhenwen@pku.edu.cn, jasperguo2013@stu.xjtu.edu.cn, liuqian@sea.com
jlou@microsoft.com, taoxie@pku.edu.cn

## Abstract

Previous research has shown that the design of Meaning Representation (MR) greatly influences model performance of a neural semantic parser. Therefore, designing a good MR is a long-term goal for semantic parsing. However, it is still an art as there is no quantitative indicator that can tell us which MR among a set of candidates may have the best final model performance. In practice, in order to select an MR, researchers often have to go through the whole training-testing process for all MR candidates, and the process often costs a lot. In this paper, we propose a data-aware metric called ISS (denoting incremental structural stability) of MRs, and demonstrate that ISS is highly correlated with model performance. The finding shows that ISS can be used as an indicator for designing MRs to avoid the costly training-testing process.

## 1 Introduction

Semantic parsing is the task of transforming a natural language utterance into a Meaning Representation (MR), whose accuracy has been significantly improved recently (Dong and Lapata, 2016; Jia and Liang, 2016; Yin and Neubig, 2017; Dong and Lapata, 2018; Iyer et al., 2017). For example, in a semantic parsing dataset of Text-to-SQL, SQL is the meaning representation.

Previous work (Guo et al., 2019; Herzig et al., 2021; Kate et al., 2005) has shown that the design of MR significantly influences model performance of a neural semantic parser. As shown in Table 1, even with the same neural network architecture, using SQL as the MR gets an execution accuracy of 61.0 on the ATIS dataset and 72.1 on the GeoQuery dataset, while using Lambda as the MR reaches 82.1 and 80.7, respectively. The difference of model performance between the MRs even reaches 20% under the same conditions.[1] The difference in model performance between MRs indicates that different MRs bring different levels of learning difficulties to a neural model.

Table 1: The execution accuracy of a neural seq2prod model for different MRs on three semantic parsing datasets: ATIS, GeoQuery, and Job. The results come from the work of Guo et al. (2020).

| Dataset | SQL | Lambda | Prolog | FunQL |
|---|---|---|---|---|
| ATIS | 61.0 | 82.1 | 79.2 | 82.7 |
| GeoQuery | 72.1 | 80.7 | 79.5 | 86.2 |
| Job | 87.6 | 91.0 | 84.4 | 92.4 |

In order to reduce learning difficulty of MRs, some researchers have put efforts to design customized MRs for their tasks. For example, Guo et al. (2019) propose a new MR named SemQL to bridge the gap between utterances and SQL. Compared with SQL, the structure of SemQL is simpler, decreasing learning difficulty of SemQL. In addition, Herzig et al. (2021) propose an MR named LIR by discarding SQL components that cannot correspond well to natural language utterances. Their experimental results show that the models can reach better model performance on LIR compared with SQL.

Despite the key role of MRs, a quantitative indicator is lacking to guide the design of easy-to-learn MRs. In order to evaluate whether a design choice of MR is good or not, researchers have to go through the whole model training-testing process iteratively (Guo et al., 2019; Herzig et al., 2021; Kate et al., 2005), incurring huge costs. Hence, it is important to find a quantitative indicator so that researchers can design MRs under a clear guideline before the model training-testing process. Guo et al.

[*] The corresponding author is Tao Xie. Jiaqi Guo and Qian Liu contributed to this work during their internships at Microsoft Research Asia.

[1] For a fair comparison, the work of Guo et al. (2020) uses the same neural network model for each MR. The hyperparameters of the model are selected by grid search for each MR.



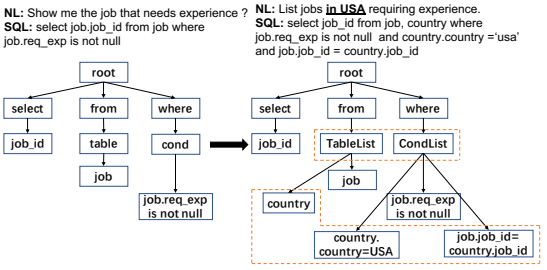
*Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3616–3625
December 7-11, 2022 

(2020) conduct an investigation on potential factors that influence learning difficulty of MRs. Their work studies three factors including the number of grammar rules, the number of production rules in an MR, and the proportion of program alias. However, none of these factors can be directly used as an indicator because each of them only partially reflects the underlying learning difficulty.

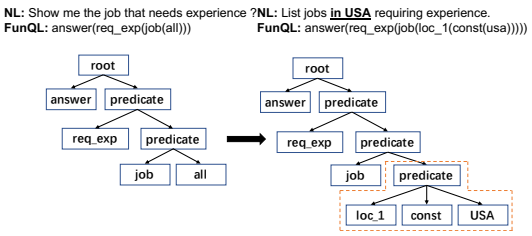
In this paper, we propose the notion of incremental structural stability and argue that it is a good quantitative indicator for learning difficulty of MRs. Incremental structural stability refers to the degree of the structural changes in an MR instance (denoted as MRI for simplicity) when its corresponding natural language utterance is changed by adding a constraint condition. Intuitively, larger changes in an MRI structure bring more challenges to neural models, leading to higher learning difficulty. As shown in Figure 1(a), the structure of SQL changes a lot when a constraint condition ("*in USA*") is added into its utterance. This simple change drastically influences the contents of the *from* clause and *where* clause of SQL. In the *where* clause, a condition for cross-table joining is added, and this condition addition is hard to predict because it depends on the database structure. In Figure 1(b), compared with SQL, FunQL (Kate et al., 2005)(the best performing MR among the four MRs in Table 1) maintains high stability of the structure when the same constraint condition is added, i.e., only adding a single predicate.

In order to measure the incremental structural stability of MRs on datasets, we propose ISS (denoting incremental structural stability), a data-aware metric. ISS quantifies the structural difference between a pair of MRIs whose corresponding utterances' semantics differ by only a constraint condition. In particular, given an utterance dataset $D$ (for a specific database schema) annotated in an MR $M$, the value of *ISS(M, D)* is computed in three steps: (1) anonymize the utterances (Dong et al., 2019); (2) extract all pairs of MRIs (in $D$) whose corresponding anonymized utterances differ by only a constraint condition; (3) measure the degree of the structural changes between each pair of MRIs.

To evaluate the effectiveness of ISS on measuring learning difficulty of MRs, we conduct experiments on the UNIMER benchmark (Guo et al., 2020). UNIMER contains three widely used semantic parsing datasets: ATIS (Price, 1990), Geo-




(a) An example of SQL




(b) An example of FunQL

Figure 1: An example of the structural changes of two MRs (SQL and FunQL), when their corresponding natural language utterances' semantics are changed by adding a constraint condition ("*in USA*"). Both Figure (a) and Figure (b) can be divided into left and right parts, each containing a natural language utterance, the corresponding MRI, and the tree structure of the MRI. The utterance in the right part with an additional constraint condition is similar to the left. The structural changes of the MRIs after adding a constraint condition are circled by dashed lines.

Query (Zelle and Mooney, 1996), and Job (Tang and Mooney, 2001), each of which is annotated in four MRs: SQL, Lambda Calculus, Prolog, and FunQL.

The experiments consist of three steps. First, we compute the ISS value for each combination of four MRs $\times$ three datasets (12 ISS values in total). Second, we measure model performance (shown in Table 1) for each preceding combination. Higher model performance achieved on combinations involving an MR indicates lower learning difficulty of this MR. Third, we compute the Pearson correlation coefficient between the values of ISS and model performance. A high absolute value of the correlation coefficient[2] shows that there is a strong correlation between ISS and learning difficulty of MRs. Our experimental results show that the correlation coefficient reaches 0.94, demonstrating that ISS is a good indicator of learning difficulty of MRs.

[2]We may not explicitly mention "absolute value" in the rest of this paper when there is no ambiguity.

In summary, this paper makes the following main contributions:

- We propose incremental structural stability as a quantitative indicator of learning difficulty of MRs.
- We propose a metric named ISS to measure the incremental structural stability of MRs. It can be used to predict model performance, decrease the experimental cost, and help researchers design easy-to-learn MRs.
- The experimental results show that the Pearson correlation coefficient between ISS and model performance is 0.94, supporting our hypothesis.

## 2 Method

We propose a data-aware metric named ISS to measure the incremental structural stability of MRs. In this section, we introduce the definition of ISS and computation of the ISS.

### 2.1 Metric Definition

Incremental structural stability is the degree of the structural changes of MRs when the corresponding natural language utterances' semantics change by adding a constraint condition. In an ideal situation, to compute ISS, given a data point $< u, m >$ from the dataset, where $u$ represents an utterance and $m$ is its corresponding MRI, we need a new data point $< u', m' >$ where $u'$ can be seen as adding a constraint condition to $u$, and $m'$ is the corresponding MRI of $u'$, and then the structural difference between $m$ and $m'$ can reflect the degree of the incremental structural stability. However, it is difficult for machines to automatically translate $u'$ to $m'$. To address this problem, instead of generating data, we traverse the dataset to search for the pairs of data whose utterances' semantics differ by only one constraint condition, and then compute the structural difference of MRIs between the pairs of data. The formulation of ISS is

$$ISS(mr, D) = \frac{1}{N} \sum_{\substack{<u,m>\in D \\ <u',m'>\in D \\ u'-u=c}} TED_{st}(ast(m), ast(m')) \quad (1)$$

where $mr$ is the given MR, $D$ is the dataset annotated in $mr$, $N$ is the number of data pairs from $D$, the utterance $u'$ can be seen as adding a constraint condition $c$ to $u$, $ast$ is the function to get the abstract syntax tree (AST) of the given MRI, and $TED_{st}$ is the function to measure the structural difference between the two given ASTs. We introduce the $TED_{st}$ function later in Section 2.2.2. An important note here is that ISS is a kind of distance, so a higher ISS reflects a lower degree of incremental structural stability.

Table 2: The definition of the *CondPart* of four MRs, along with an example that the *CondPart* corresponds to a constraint condition of destination in the utterance.

| MR | CondPart | Example |
|---|---|---|
| SQL | condition clause except table joining | city.city_name =value |
| Lambda | function | _to $0 value:_ci |
| Prolog | object | city_name(value) |
| FunQL | predicate | city_name(value) |

### 2.2 Computation Procedure

The computation procedure of ISS consists of three steps.

**Step 1**. We anonymize all MRIs in dataset $D$ by replacing all concrete entities and numbers with their data types. Currently, the supported data types include string, number, and DateTime types (e.g., date, year, month). By anonymization, the algorithm mainly focuses on the structure of MRIs rather than the concrete domain-specific values.

**Step 2**. We get all anonymized pairs from dataset $D$ such that the utterances' semantics in each pair differ by only one constraint condition.

**Step 3**. We measure the structural difference between the MRIs of the data pairs using Eq1. The details are presented in the subsequent two subsections.

#### 2.2.1 Attainment of Data Pairs

Due to the complexity of natural language, it is difficult to analyze natural language for judging whether two utterances' semantics differ by one constraint condition. We instead analyze the corresponding MRIs of utterances and utilize the abstract syntax tree (AST) of MRIs to get the required data pairs from the dataset. There exists a clear mapping between the constraint conditions in an utterance and the specific components of MRs. We name the specific components of MRs as *CondPart*, parts of which are listed in Table 2. For example, in SQL, each condition clause except the conditions only relevant for joining tables is a $CondPart$. In Lambda Calculus, each function is a $CondPart$. In Prolog and FunQL,

**Algorithm 1:** Get Data Pairs

**Input:** $D \longrightarrow$ Dataset containing utterances and MRIs
**Input:** $\delta \longrightarrow$ Threshold of difference of *CondParts*
**Input:** $K \longrightarrow$ Threshold used to filter MRI pairs
$T \leftarrow \{\}$; Storing all MRIs and their ASTs
**for** $u, m$ *in* $D$ **do**
  $t = ast(m)$;
  $t = anonymize(t)$;
  $T = T \cup \{< m, t >\}$;
**end**
$M \leftarrow \{\}$; A mapping that stores candidate data pairs according to the difference of *CondParts*
**for** $m$, $t$ *in* $T$ **do**
  **for** $m'$, $t'$ *in* $T \backslash \{< m, t >\}$ **do**
    $condSet = getCondPart(t)$;
    $condSet' = getCondPart(t')$;
    **if** $condSet \subset condSet'$ **then**
      $condDiff = condSet' - condSet$;
      **if** $size(condDiff) \leq \delta$ **then**
        $M[condDiff] = M[condDiff] \cup \{< (m,t),(m',t') >\}$;
      **end**
    **end**
  **end**
**end**
$P \leftarrow \{\}$; Storing required data pairs
**for** $V$ *in* $M.values()$ **do**
  **if** $size(V) \geq K$ **then**
    $P = P \cup V$
  **end**
**end**
return $P$;

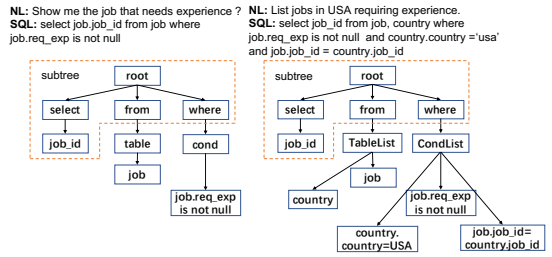
each predicate is a $CondPart$. It is worth noting that a single constraint condition in an utterance may correspond to multiple *CondParts* in the generated MRI. For example, a time constraint condition in an utterance can correspond to two *CondParts*: $time.month = value$ and $time.day = value$. Compared with constraint conditions in utterances, the *CondParts* of MRIs can be easily extracted.

Based on the preceding observation, we propose an algorithm that utilizes the ASTs of MRIs to first get the *CondParts* of MRIs, and then get pairs of data whose utterances' semantics differ by one constraint condition by comparing the *CondParts* of MRIs. Algorithm 1 describes the detail of getting the required data pairs. As a data-aware algorithm, given a dataset $D$ annotated in the MR $mr$, and the hyper-parameters $\delta$ and $K$, the algorithm returns all the required data pairs to be used to compute $ISS(mr, D)$. First, all of the MRIs from $D$ are parsed to get their corresponding ASTs. Second, considering that some *CondParts* such as $Country.country =$ "$USA$" and $Country.country =$ "$UK$" should be considered the same because they have the same structure in ASTs, and they correspond to the same type of constraint condition in utterances, the algorithm anonymizes all of the ASTs to eliminate the impact of values. In the $anonymize$ function, the ASTs are traversed and the nodes labeled by specific numbers or strings are replaced by new nodes labeled as their types (Number or String). Finally, the algorithm enumerates all the MRIs and their ASTs to get the required data pairs in three steps:

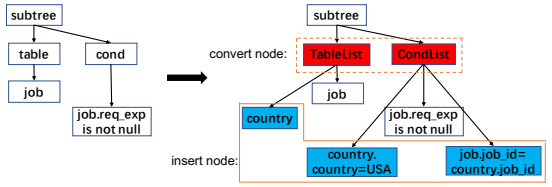
1. Utilize ASTs to get the sets of *CondParts* in the MRIs. The $getCondPart$ function in Algorithm 1 can get the set of *CondParts* based on the used MR.

2. As mentioned earlier, a single constraint condition in an utterance may correspond to multiple *CondParts* in the generated MRI. To handle such cases, the algorithm selects candidate pairs (from the dataset) where the difference between two MRIs of each pair is multiple CondParts, and the number of the CondParts in the difference is no more than $\delta$ (here, $\delta$ is a constant threshold, whose value relies on only the used MR).

3. However, the candidates obtained from the preceding step may contain data pairs that the utterances' semantics differ by more than one constraint condition. For example, when the $\delta$ equals two, two SQL queries differ by two CondParts, $departure =$ "$USA$" and $destination =$ "$China$", can be selected as a candidate. But their corresponding utterances' semantics differ by two constraint conditions (departure and destination). At this step, the algorithm tries to filter out these candidates. In reality, it is natural that the frequency of an utterance difference containing only one constraint condition is usually much




(a) A pair of data that the utterances' semantics differ by one constraint condition. The common subtree (circled by dashed lines) of these two ASTs is replaced by a new node labeled as $subtree$.




(b) After subtrees with nodes are replaced, the value of TED is computed. The value of TED between these two ASTs is five, with two converted red nodes (circled by dashed lines) and three inserted blue nodes (circled by solid lines).

Figure 2: An example of the $TED_{st}$ process.

higher than the frequency of a difference containing more than one constraint condition, because most real-world queries involve only no more than three conditions. Based on this insight, the algorithm stores the candidates into a mapping according to their difference in *CondParts*, and then excludes the candidates that the frequency of their difference in *CondParts* is less than a threshold $K$. This operation helps us filter out the noisy ones and make sure that most remaining pairs have a single constraint condition in each pair's utterance difference.

#### 2.2.2 Measurement of Difference

After getting the data pairs that contain MRIs and their corresponding ASTs, we compute ISS by measuring the average structural difference between the pairs. We use the tree edit distance (TED) that is computed over the ASTs of the MRIs to measure the structural difference. Given two ASTs $t_1$ and $t_2$, TED refers to the minimum number of edit operations to transform $t_1$ to $t_2$. There are three edit operations: insert a node into the tree, delete a node in the tree, and convert the label of a node into another. There is a lot of work on TED (Tai, 1979; Klein, 1998; Demaine et al., 2010). We use the algorithm proposed by Pawlik and Augsten (2020) to compute the TED of $t_1$ and $t_2$.

Because the granularity of nodes in ASTs will influence the size of ASTs and the value of TED, and different MRs often have different node granularities, it is unfair to use the value of TED to measure different MRs whose node granularities are different. Unlike a node, a subtree is a better unit for edit operations in our scenario, because a subtree is less affected by the granularity of a node. Inspired by the work of Iyer et al. (2019), given a data pair, we traverse the pair of two ASTs $t_1$ and $t_2$ to find all of the common subtrees that appear in both $t_1$ and $t_2$. Then we replace each subtree by a new node that represents the entire subtree. Finally, we compute the value of TED named $TED_{st}$ based on the pair of ASTs with subtrees replaced.

As shown in Figure 2, we compute the value of $TED_{st}$ in two steps: (1) replace all of the common subtrees in ASTs; (2) get the value of TED based on the pair of ASTs with subtrees replaced.

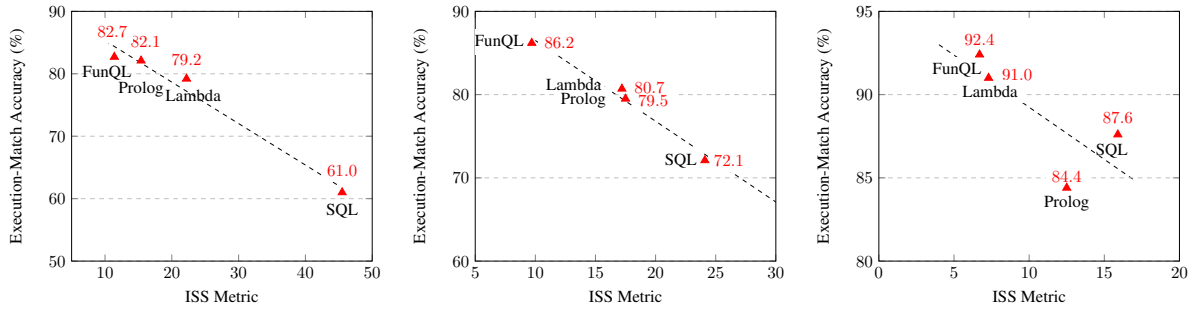
## 3 Experimental Setup

In this section, we introduce the details of our experiments. We present the used datasets and the metric for evaluating ISS. We also present the implementation details of ISS and two baselines used to compare with ISS.

### 3.1 Dataset and Evaluation

We conduct experiments on the UNIMER benchmark (Guo et al., 2020). To our best knowledge, UNIMER is the benchmark with the most types of MRs annotated. UNIMER contains three widely used domain-specific datasets: ATIS, GeoQuery, and Job, each annotated in four MRs: SQL, Lambda Calculus, Prolog, and FunQL. We use the template split of these three datasets. Because the data is independent and identically distributed under the template split, only the training data is used to estimate ISS.

Following Guo et al. (2020), for each pair of the dataset and MR, we consider two neural semantic parsing models for generating MRIs, namely Seq2Prod (Dong and Lapata, 2016) and Seq2Seq (Yin and Neubig, 2017), because many neural parsers are developed based on them. Model performance is evaluated with both the exact-match accuracy and execution-match accuracy, which are two widely used metrics to measure model performance. When using the former, we consider the prediction of the model to be correct when the pre-




(a) ATIS (b) GeoQuery (c) Job

Figure 3: The values of ISS on 4 MRs × 3 datasets.

diction exactly matches (string match) the ground truth. When using the latter, we consider the prediction of the model to be correct when the execution result (query result) of the prediction matches that of the ground truth.

To evaluate the effectiveness of ISS, we adopt the Pearson correlation coefficient to measure the correlation between model performance and the values of ISS. Specifically, since there are 12 pairs of MR and dataset (four MRs × three datasets), we compute the absolute values of Pearson coefficients over 12 pairs of model performance and ISS values. A higher absolute value of the Pearson correlation coefficient indicates that the metric has a stronger capability to reflect learning difficulty of MRs on the given datasets.

### 3.2 Algorithm Configurations

Based on our observation, a constraint condition in an utterance typically corresponds to no more than three *CondParts* in an MRI, so for all four MRs we set the $\delta$ hyper-parameter to the value of three. Based on the size of the dataset, we set the $K$ hyper-parameter the values of five, three, and two for the ATIS, GeoQuery, and Job datasets, respectively. Based on our manual inspection, Algorithm 1 can filter out the candidate data pairs that do not meet the requirement under these values of $K$.

### 3.3 Baseline Metrics

For comparison purposes, we also consider the following two metrics as baselines.

**Depth of AST.** The average depth of MRIs can reflect the difficulty of neural network models in the stage of decoding. Deep ASTs can increase the difficulty of model learning. Kwiatkowksi et al. (2010) claim that the depth of ASTs can explain the difference in learning difficulty to some extent.

Table 3: The absolute Pearson correlation coefficients between the three metrics and model performance. For each metric, we compute 12 values (four MRs × three datasets), and for each model, there are also 12 corresponding values of model performance.

| Model | Seq2Prod | | Seq2Seq | |
|---|---|---|---|---|
| | Exact | Execution | Exact | Execution |
| ISS | **0.809** | **0.940** | **0.511** | **0.862** |
| Depth | 0.121 | 0.608 | 0.305 | 0.487 |
| Size | 0.363 | 0.783 | 0.382 | 0.744 |

They believe that the deeply nested structure of an MR makes it more challenging to be learned. Formally, given a dataset $D$ annotated in an MR denoted as $mr$, the average depth of ASTs can be computed by

$$Depth(mr, D) = \frac{1}{N} \sum_{<u,m>\in D} depth(ast(m)) \quad (2)$$

where $N$ is the size of $D$, $u$ is an utterance in $D$, $m$ is the corresponding MRI, $ast$ is the function that gets the AST of $m$, and $depth$ is the tree depth of an AST.

**Size of AST.** The size of an AST is the number of nodes in the AST. This size can partially reflect the complexity of MR structure. Guo et al. (2020) claim that the number of grammar rules in an MRI, i.e., the size of an AST, affects learning difficulty of the used MR. Given a dataset $D$ annotated in $mr$, the average size of ASTs can be computed by

$$Size(mr, D) = \frac{1}{N} \sum_{<m,u>\in D} size(ast(m)) \quad (3)$$

where $size$ is the number of nodes (grammar rules) of an AST.

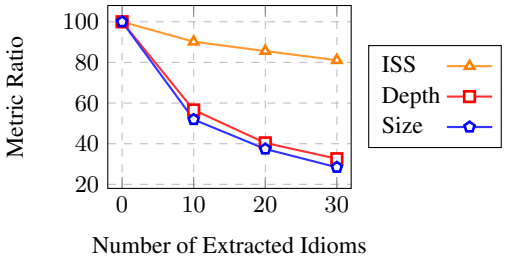



Figure 4: The metric values of ISS(SQL, ATIS), Depth(SQL, ATIS), and Size(SQL, ATIS) when grammars are changed by idiom extraction. The y-axis shows the ratio of the values of the metrics for the new grammars to the original grammar.

## 4 Results and Discussion

In this section, we present the experimental results to show the effectiveness, robustness, and low cost of ISS, respectively.

### 4.1 Correlation between Model Performance and ISS

In this subsection, we compare three metrics: our ISS, the *Depth* of AST (in short as *Depth*), and the *Size* of AST (in short as *Size*); the last two serve as baselines, being used to measure the complexity of an MR grammar in previous work (Guo et al., 2020; Kwiatkowksi et al., 2010). Table 3 shows the absolute values of Pearson correlation coefficients between model performance and the three metrics, respectively. Compared with the two baseline metrics, ISS has the highest absolute correlation coefficients with model performance for two models (Seq2Prod and Seq2Seq) measured with two metrics (exact-match and execution-match accuracy), respectively. For example, the absolute correlation coefficient between ISS and the execution-match accuracy of Seq2Prod reaches 0.94. The results show that our metric reflects learning difficulty of MRs on different datasets.

Figure 3 shows the results of ISS on four MRs $\times$ three datasets. The results show that there is a strong linear correlation between ISS and model performance. According to the results, it can be concluded that the values of ISS reflect learning difficulties of different MRs. In other words, ISS can be used to predict model performance and guide the designing of easy-to-learn MRs.

### 4.2 Robustness on MR Grammar Changes

In this subsection, we further study the behaviors of ISS and the baseline metrics when the grammar of a single MR changes. In this paper, we use a technique called idiom extraction to manipulate MR grammars; this technique is proposed to improve the performance of a neural semantic parser by changing the grammar of an MR (Iyer et al., 2019). We follow the work of Iyer et al. (2019) to extract idioms from the SQL grammar to generate a new idiom-aided SQL grammar. According to the definition, the $Depth$ and $Size$ metrics can be greatly influenced by changing the grammar of an MR. The results are shown in Figure 4, where we present the ratios of different metrics for the new SQL grammars to the original grammar on the ATIS dataset. We can see that, after idiom extraction, there is a sharp decrease in the average values of $Depth$ and $Size$. However, in our experiment, model performance is only slightly improved when 30 idioms are extracted (e.g., the accuracy changes from 0.65 to 0.67), while the values of $Depth$ and $Size$ decrease to about $\frac{1}{3}$ of the original values. The experimental results show that these two baseline metrics are not able to robustly reflect learning difficulty of MRs. On the other hand, the ISS value is only slightly affected by idiom extraction. In other words, ISS can still robustly reflect learning difficulty. Based on the experiment results, we can conclude that ISS is a much better indicator than the baseline metrics under the idiom extraction.

### 4.3 Computation Process Analysis

Further, we conduct a thorough analysis of the process of getting data pairs, because getting data pairs is the key process in computing ISS. Table 4 shows two examples of pairs of data whose utterances' semantics differ by one constraint condition. The examples show that our algorithm gets the required data pairs from the dataset. The "nonstop" constraint conditions of the two utterances are *Nonstop* and *I don't want any stopovers*, respectively. Although the form of the "nonstop" constraint condition in utterances varies, our algorithm gets data pairs because of the utilization of ASTs.

Since the computation of ISS relies on the data pairs extracted by the algorithm from the dataset, we count the data coverage of the datasets. Data coverage is the proportion of the data used when computing ISS in the dataset. The results are shown in Table 5. As shown in the table, the coverage of all MRs on datasets is greater than 50%. High data coverage indicates that the computation of ISS uses most of the data in the dataset, and this result also

Table 4: Two constraint conditions in utterances: airline constraint and nonstop constraint in ATIS. For each constraint condition, two pairs of sentences are listed as Example 1 and Example 2. The utterances' semantics in the examples differ by one constraint condition. The constraint conditions are underlined.

| constraint condition | Example 1 | Example 2 |
|---|---|---|
| Airline constraint | **Sent1**: List all AA from Milwaukee to Phoenix on Saturday.<br>**Sent2**: What flights from Tacoma to Orlando on Saturday? | **Sent1**: List round trip flights between Boston and Oakland using TW.<br>**Sent2**: List all round trip flights from Orlando to Kansas CITY. |
| Nonstop constraint | **Sent1**: Nonstop flights from New York city to Las Vegas on Sunday.<br>**Sent2**: What flights from Tacoma to Orlando on Saturday? | **Sent1**: I want to fly from Boston to Denver and I don't want any stopovers and I'd like to fly only during the afternoon.<br>**Sent2**: From Las Vegas to Phoenix departing in the morning. |

Table 5: The coverage of the data used in the computation process of ISS.

| Dataset | SQL | Lambda | Prolog | FunQL |
|---|---|---|---|---|
| ATIS | 0.82 | 0.80 | 0.81 | 0.81 |
| GeoQuery | 0.74 | 0.73 | 0.65 | 0.68 |
| Job | 0.60 | 0.62 | 0.54 | 0.60 |

Table 6: The time cost of computing ISS (the unit of time is seconds).

| Dataset | SQL | Lambda | Prolog | FunQL |
|---|---|---|---|---|
| ATIS | 167.79 | 46.09 | 80.12 | 19.47 |
| GeoQuery | 4.62 | 3.14 | 2.96 | 1.07 |
| Job | 0.48 | 0.31 | 0.49 | 0.25 |

supports that ISS reflects learning difficulty of MRs on datasets.

### 4.4 Time Cost

One advantage of ISS is that the time cost of computing ISS is much less than training a model. As shown in Table 6, the values of ISS are computed within three minutes using a single-threaded program on a desktop machine with Intel CPU at 2.8GHz without any GPU resource. The time cost changes as the size of datasets and the average size of ASTs vary. As a comparison, to get the performance of a neural parser, researchers need to train a DNN model multiple times to search for hyper-parameter values, and each time often takes a few hours. Therefore, using ISS to predict the performance of a neural parser can help practitioners significantly reduce the cost of experiments when they are designing easy-to-learn MRs for semantic parsing.

## 5 Related Work

MRs play an important role in semantic parsing. Researchers have made a lot of efforts on designing new MRs in order to improve model performance. Variants of typed Lambda Calculus are designed to outperform Lambda Calculus (Carpenter, 1997; Zettlemoyer and Collins, 2005). Liang (2013) designs DCS for querying knowledge bases. Kate et al. (2005) propose the FunQL MR, which performs better than SQL on domain-specific datasets.

There are also research efforts on decreasing learning difficulty of SQL. Guo et al. (2019) propose SemQL to bridge the gap between natural language and SQL, and achieve better performance on the Spider benchmark (Yu et al., 2018). Herzig et al. (2021) propose RIR and LIR based on SQL, improving model performance on domain-specific datasets. RIR and LIR also improve the compositional generalization of SQL. $SQL^{UF}$ (Suhr et al., 2020) and an extension of relational algebra (Rubin and Berant, 2020) are also designed to address the compositional generalization challenge of SQL.

However, the preceding MRs are usually designed based on a hunch or trial-and-error. As the final goal, designing easy-to-learn MRs is so abstract that there is no standard guideline for researchers during the design process. The motivation of designing SemQL is to align the natural language utterance with the MR, while RIR and LIR are designed to reduce the mismatch between MRIs and natural language, but the designers do not explain their motivation. In this paper, we argue that incremental structural stability could be one of the important guidelines for designing MRs. Our experimental results also support our hypothesis.

Previous work has proposed some hypotheses on which characteristics of MRs are related to learning difficulty. Kwiatkowksi et al. (2010) show that Lambda Calculus outperforms FunQL on the Geo-

Query dataset, and use the depth of ASTs to explain this phenomenon. Guo et al. (2020) propose the UNIMER benchmark and compare learning difficulty of the four MRs under a fair condition. They guess that the number of production rules and the size of ASTs are factors leading to the difference in model performance. However, their experimental results do not support their hypothesis. As a result, to the best of our knowledge, we are the first to propose a quantitative indicator that can reflect learning difficulty of MRs and can be used to guide researchers on designing MRs.

## 6 Conclusion

In this paper, we have studied the difference of MRs in order to find a quantitative value to indicate learning difficulty of MRs. We have proposed the notion of incremental structural stability and argued that it is a reasonable indicator for learning difficulty of MRs. We have proposed a data-aware metric named ISS to measure the incremental structural stability of MRs on the given datasets. We have conducted experiments to show the high correlation between ISS and model performance of the neural semantic parser.

Our findings have important implications for future work. For example, our experimental results have demonstrated that ISS can be used to guide the design of new MRs. Researchers can design an MR to achieve better parsing accuracy by improving the incremental structural stability of the MR. More importantly, the ISS-based process of designing MRs helps researchers largely avoid the high cost brought by the iterative training and testing process.

## Limitation

Our work mainly contains two limitations. First, the correlation between our ISS metric and model performance cannot imply causation, but it is still significant progress in exploring the characteristics influencing learning difficulty of MRs. Second, the choice of hyper-parameters $\delta$ and $K$ is based on human experience and observations of the datasets. The values of the ISS metric remain stable when the values of hyper-parameters vary in a reasonable range, so the choice of hyper-parameters does not influence the conclusion.

## Acknowledgement

We would like to thank all the anonymous reviewers for their constructive feedback and useful comments. Tao Xie is also affiliated with the Key Laboratory of High Confidence Software Technologies (Peking University), Ministry of Education China. This work was partially supported by National Natural Science Foundation of China under Grant No. 62161146003, and the Tencent Foundation/XPLORER PRIZE.

## References

Bob Carpenter. 1997. *Type-logical semantics*. MIT press.

Erik D. Demaine, Shay Mozes, Benjamin Rossman, and Oren Weimann. 2010. An optimal decomposition algorithm for tree edit distance. *ACM Trans. Algorithms*, 6(1).

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.

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

Zhen Dong, Shizhao Sun, Hongzhi Liu, Jian-Guang Lou, and Dongmei Zhang. 2019. Data-anonymous encoding for text-to-SQL generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5405–5414.

Jiaqi Guo, Qian Liu, Jian-Guang Lou, Zhenwen Li, Xueqing Liu, Tao Xie, and Ting Liu. 2020. Benchmarking meaning representations in neural semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1540.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-SQL in cross-domain database with intermediate representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535.

Jonathan Herzig, Peter Shaw, Ming-Wei Chang, Kelvin Guu, Panupong Pasupat, and Yuan Zhang. 2021. Unlocking compositional generalization in pre-trained models using intermediate representations. *CoRR*, abs/2104.07478.

Srinivasan Iyer, Alvin Cheung, and Luke Zettlemoyer. 2019. Learning programmatic idioms for scalable

semantic parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5426–5435.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963–973.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22.

Rohit J. Kate, Yuk Wah Wong, and Raymond J. Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 3*, page 1062–1068.

Philip N. Klein. 1998. Computing the edit-distance between unrooted ordered trees. In *Proceedings of the 6th Annual European Symposium on Algorithms*, pages 91–102.

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

Percy Liang. 2013. Lambda dependency-based compositional semantics. *CoRR*, abs/1309.4408.

Mateusz Pawlik and Nikolaus Augsten. 2020. *Minimal edit-based diffs for large trees*, page 1225–1234. ACM.

P. J. Price. 1990. Evaluation of spoken language systems: The ATIS domain. In *Proceedings of the Workshop on Speech and Natural Language*, page 91–95.

Ohad Rubin and Jonathan Berant. 2020. SmBoP: Semi-autoregressive bottom-up semantic parsing. *CoRR*, abs/2010.12412.

Alane Suhr, Ming-Wei Chang, Peter Shaw, and Kenton Lee. 2020. Exploring unexplored generalization challenges for cross-database semantic parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8372–8388.

Kuo-Chung Tai. 1979. The tree-to-tree correction problem. *J. ACM*, 26(3):422–433.

Lappoon R. Tang and Raymond J. Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proceedings of the 12th European Conference on Machine Learning*, page 466–477.

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

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.

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

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the 21th Conference on Uncertainty in Artificial Intelligence*, page 658–666.