Logic-Consistency Text Generation from Semantic Parses

Chang Shu[1*], **Yusen Zhang**[2*], **Xiangyu Dong**[3], **Peng Shi**[4], **Tao Yu**[5], **Rui Zhang**[2]

[1]School of Informatics, University of Edinburgh
[2]Department of Computer Science and Engineering, Penn State University
[3]School of Computer Science and Engineering, Beihang University
[4]David R. Cheriton School of Computer Science, University of Waterloo
[5]Department of Computer Science, The University of Hong Kong
s1783039@ed.ac.uk, yfz5488@psu.edu, dxy912434027@gmail.com
peng.shi@uwaterloo.ca, rmz5227@psu.edu

Abstract

Text generation from semantic parses is to generate textual descriptions for formal representation inputs such as logic forms and SQL queries. This is challenging due to two reasons: (1) the complex and intensive inner logic with the data scarcity constraint, (2) the lack of automatic evaluation metrics for logic consistency. To address these two challenges, this paper first proposes SNOWBALL, a framework for logic consistent text generation from semantic parses that employs an iterative training procedure by recursively augmenting the training set with quality control. Second, we propose a novel automatic metric, BLEC, for evaluating the logical consistency between the semantic parses and generated texts. The experimental results on two benchmark datasets, *Logic2Text* and *Spider*, demonstrate the SNOWBALL framework enhances the logic consistency on both BLEC and human evaluation. Furthermore, our statistical analysis reveals that BLEC is more logically consistent with human evaluation than general-purpose automatic metrics including BLEU, ROUGE and, BLEURT. Our data and code are available at https://github.com/Ciaranshu/relogic.

1 Introduction

Natural language generation (NLG) from semantic parses is to generate the text description for the formal representation input such as logical forms, AMR, and SQL queries. It has drawn widespread attention because of its substantial contributions to the interpretability and usability of the latest natural language interfaces (Gatt and Krahmer, 2018; Chen et al., 2020b; Hu et al., 2020; Mishra et al., 2019; Yu et al., 2019; Ngomo et al., 2013; Wang et al., 2018; Gardent et al., 2017; Wang et al., 2020a; Wang, 2019; Koutrika et al., 2010a). Recently, pre-trained large-scale language models like BERT (Devlin et al., 2018), T5 (Raffel et al., 2020), and GPT-3 (Brown et al., 2020) have raised the ability to generate natural language from formal texts to a promising level of fluency and coherence.

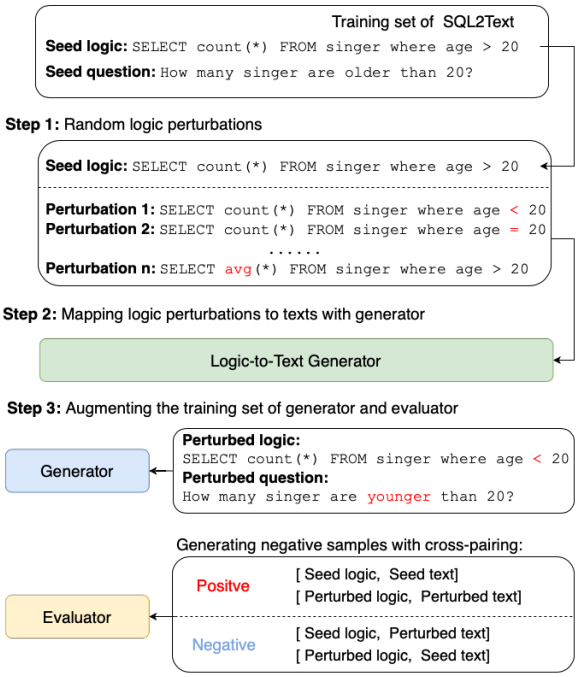

Figure 1: Our data augmentation procedure for the generator and evaluator in the SNOWBALL framework.

However, NLG from semantic parses still has suffered from two crucial challenges: (1) the data scarcity constraint due to the bias on certain types of logic forms or expensive labeling work (Iyer et al., 2017; Yaghmazadeh et al., 2017), which potentially leads to the unsatisfied fidelity of remaining the complex and intensive inner logic in the generated text based on our empirical research; (2) The general-purpose automatic metrics (Novikova et al., 2017a) such as BLEU (Papineni et al., 2002), ROUGE (Lin, 2004) and BLEURT (Sellam et al.,

*Equal Contribution

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4414–4426
August 1–6, 2021.

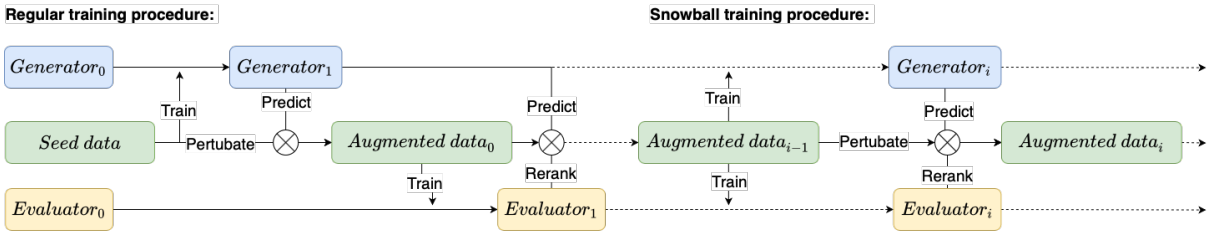

Figure 2: Our SNOWBALL framework employs an iterative training procedure over a generator and evaluator through data augmentation.

2020) are not ideal for explicitly measuring the logic consistency (Wang et al., 2020b; Harkous et al., 2020), because they tend to evenly weight each word in the generated text without fully attending on the fatal logical keywords.

To address these two critical problems, we propose the SNOWBALL framework for high-fidelity text generation from semantic parses and the BLEC automatic evaluation metric for logic consistency:

Snowball Framework. Our SNOWBALL framework, as illustrated in Figure 2, trains two modules to ensure high-fidelity text generation: (1) a generator that maps the logical form to its textual description, and (2) an evaluator that indicates the logic consistent score of each pair of logical form and textual sentence. Rather than training the generator and evaluator independently, SNOWBALL performs iterative training on the generator and the evaluator. To deal with the data scarcity issue, we propose a data augmentation procedure to cover valid logic variations with diverse natural language expressions to improve generalizability. To this end, during each iteration, various unseen logic pairs could be automatically generated with rule-based enumerated logic forms and their corresponding text predicted by the generator. The evaluator is then used to filter out the high reliable augmented logic pairs for the next training iteration.

BLEC Metric. To evaluate the logic consistency of the text generated by the model, we propose a rule-based automatic evaluation metric called Bidirectional Logic Evaluation of Consistency, or BLEC. It takes the logical form and the generated corresponding natural language text as input, then outputs a label indicating if they represent consistent logic. Compared with the neural network evaluator, BLEC can be easily deployed to different datasets, as long as the parser (i.e., the grammar of the logical form) is given.

In our experiments, we exam the effectiveness of our proposed approaches on the benchmark datasets of NLG from semantic parses derived from existed Text-to-SQL dataset *Spider* (Yu et al., 2018) and Table-to-Text dataset *Logic2Text* Chen et al. (2020b). Our analysis shows that our BLEC metric has a substantially positive Pearson score with human annotations, demonstrating better logic consistency than other automatic metrics. The BLEC result shows that the SNOWBALL framework leads to accordant enhancement in logic consistency on two datasets compared to the single-pass training method based on BART (Lewis et al., 2020).

Our key contributions are summarized into threefolds: (1) We propose a simple but effective training framework SNOWBALL that strengthens the logic faithfulness of generated text by covering diverse logic variations. (2) We propose a new logic evaluation metric BLEC that accurately measures the logical consistency with a refined keyword matching mechanism. (3) Our experiment results demonstrate that SNOWBALL at most increases the BLEC from 10.1% on SQL-to-Text and 1.2% on Logic-to-Text tasks compared to the baseline. Moreover, our statistical analysis reveals that BLEC achieves a +0.66 Pearson correlation coefficient compared with human labels, serving as a much better automatic evaluation metric than not only the traditional BLEU and ROUGE metrics, but the latest BLEURT metrics.

2 Related Work

2.1 Parses-to-Text

The source of data-to-text (D2T) datasets is mostly a flat ontology structure, like E2E(Novikova et al., 2017b), LogicNLP(Chen et al., 2020a), RotoWire(Wiseman et al., 2017), and ToTTo(Parikh et al., 2020), which is not powerful enough to encode rich semantic relationships in the ontology. Second, some datasets, such as WebNLG(Gardent et al., 2017), E2E, and RotoWire, have a limited number of domains. E2E is on the restaurant domain, and RotoWire is on the basketball domain.

Moreover, some of them only have loose alignments between input and sentence, e.g., RotoWire.

Generating the natural language descriptions for the logic forms or parses as a sub-task of D2T, has been studied in various datasets and tasks, such as GCC grammar to text (White, 2006), and UCC grammar to text (Gardent and Plainfossé, 1990). There are a lot of works that leverage the neural networks to conduct the generation on various tasks, for example, generating natural language from AMR (Song et al., 2018; Ribeiro et al., 2019; Damonte and Cohen, 2019), logic forms (Chen et al., 2020b), as well as SQL parses (Xu et al., 2018; Ngonga Ngomo et al., 2013; Koutrika et al., 2010b). However, different from these works, our work focuses on the logic consistency generation from parses. So we will mainly discuss and evaluate the model based on the logic between parses and questions.

2.2 High-fidelity Text Generation

As for the end-to-end neural-based text generation models, collaborating the auxiliary task during model training is an intuitive method that introduces the logic regulation to the models. For instance, the fidelity classification task proposed by Harkous et al. (2020), the auxiliary span extraction tasks by Kryscinski et al. (2020),the table-text optimal-transport matching and embedding similarity losses by Wang et al. (2020b) and the content matching task presented by Parikh et al. (2020) are proved to be effective. Nevertheless, to the best of our knowledge, we are the first to bridge the training procedure of evaluator and generator together with the iterative training framework snowball. Furthermore, we attempt to construct a new automatic metric and a new dataset dedicated to evaluating the logic consistency of text generation. The concentration of our work differs from the related high-fidelity text generation work (Chen et al., 2020b; Chan et al., 2019; Nie et al., 2018; Tian et al., 2019; Wang et al., 2020a), by attempting to present the panorama of the challenges of logic-consistent text generation instead of focusing on the model-wised modifications.

3 Snowball Framework

The SNOWBALL framework addresses the challenge of the complex and intensive inner logic with data sparsity constraint for the high-fidelity text-generation from semantic parses. As illustrated in Figure 2, SNOWBALL assures the logic consistency with three bases: (1) Iterative training procedure synergistically enhances the generator and evaluator in the adversarial fashion; (2) Data augmentation based on rule-based logic perturbations and neural-based text generation covering diverse unseen logic variations for iterative training; (3) Structure-aware encoding boost the sensibility of the encoder on mild logic shift.

3.1 Iterative Training

Rather than training the generator and evaluator independently, SNOWBALL performs training on the generator and the evaluator iteratively. As demonstrated in Figure 2, the prerequisite of the snowball training procedure is the regular training procedure: (1) the $Generator_0$ is trained on the benchmark NLG datasets with the normal end-to-end approach into trained $Generator_1$; (2) meanwhile, the logic forms in the seed data are converted into variations with given rules, then the $Generator_1$ predicts the text for each mutated logic forms to be a completed logic pair; (3) The initial $Evaluator_0$ is then trained on those augmented logic pairs.

Then, during the SNOWBALL procedure, the generator and evaluator are collaboratively improved through several training iterations, and during each iteration, a three-step adversarial interaction would be conducted between the generator and evaluator: **Step 1:** The trained $Evaluator_{i-1}$ could be used to rerank the beam search results given by the decoder of the generator, consequently leading to increased quality of the augmented logic pairs, $Augmented\ data_{i-1}$; **Step 2:** The $Generator_i$ is capable to better retain the logic consistency by training on the $Augmented\ data_{i-1}$ which contains more unseen logic variations uncovered in the seed data; **Step 3:** The enhanced $Generator_i$ predicts the increasingly realistic-like perturbed sentences from the perturbed logical forms, which brings more challenging negative samples to the training set of the $Evaluator_i$. The data augmentation in the first step would be further described in Section 3.2.

To be specific, our generator and evaluator in SNOWBALL are described as follows.

Generator The generator maps the logical form to the corresponding natural language sentences. We choose the pre-trained BART model (Lewis et al., 2020) following the standard transformer architecture (Vaswani et al., 2017), which contains

the encoder and decoder architecture as the denoising autoencoder pre-trained on the task of corrupted text reconstruction. The input of the encoder is the structure-aware representation of the logic forms (Section 3.2), while the target output of the decoder is the aligned textual description for the input parses.

Evaluator An evaluator indicates the logic consistent score of pairs of logical forms and textual sentences, which is vital for assessing the performance of the logic-focused text generator. In contrast to other text generation tasks, generating sentences from logical forms especially requires the evaluator to be reasonably sensitive to the subtle logic shifts of the model predictions. For instance, deleting negation words such as 'not' is fatal for our task by significantly compromising the logic consistency. Therefore, we exploit a binary classification architecture similar to the BART-based natural language inference model (Lewis et al., 2020) as our evaluator to compute the consistency between the pairs of logical form and text $[L, Q]$. The input of the encoder is the concatenation of the L and Q appended an $[EOS]$ token, and the logic scores γ are computed as:

$$\gamma = \sigma(\omega([h_{d_1}, h_{d_2}, h_{d_3}...])) \quad (1)$$

where h_{d_n} denotes the last hidden states of the decoder, ω denotes the max-pooling layer, and σ is the sigmoid activation function.

3.2 Data Augmentation

As the labeled training data for both the generator and evaluator is extremely limited, we propose a data augmentation procedure to enlarge the training set by covering variations of logic forms paired with diverse natural language expressions to improve the generalizability. To be specific, our data augmentation consists of three steps as depicted in Figure 1 from a seed dataset with human annotation:

Step 1: Logic perturbation Instead of modifying the natural language sentences, we choose to corrupt the logic consistency by perturbing logical forms mainly because of two reasons: (1) The regular structures of logical forms guarantee the procedure of the logical corruption to be comparatively controllable; (2) The perturbed logical forms could be easily validated with the corresponding parser and grammar checker. The perturbations of each given logical form could be enumerated exhaustively according to hand-tuned rules to cover the following logic inconsistencies:

- **Logic shift:** The logic shift indicates that the generated text logically distinct from the input logical forms, such as turning the assertive sentences into negative sentences. This could be attributed to the perturbations of aggregators, operators, logic conjunction, etc.
- **Phrase and number changes:** The phrase changes mean that the generated sentence modifies the appointed phrase from the logical forms, while the number changes are that the numerical values in the logical forms are perturbed.
- **Entity insertion, deletion and swapping:** Perturbations of entities is a common drawback that most natural language generation models suffer. This includes the phenomenon that the predicted sentences neglect the entities mentioned in the logical forms, insert unrelated entities to the logical form, or mislay them.

Step 2: Inference from perturbed logic After logic perturbation, the generator could be exploited as the artificial annotator to generate the corresponding sentence for each logical form in a semi-supervised manner. Compared to the rule-based or template-based method, the recent pre-trained seq-to-seq models empirically generate the natural language sentences with better fluency and coherency. Though this method could easily create a considerable amount of labeled data meanwhile avoid the expensive human annotation, what can not be ignored is that the model-based generator naturally would introduce unexpected noise during augmentation. Therefore, the quality control for the data augmentation is one of the most crucial cornerstones for a satisfactory result.

Step 3: Dataset composition As shown in Figure 1, the example in the seed dataset is denoted as [Seed logic, Seed text], and the augmented examples are denoted as [Perturbed logic, Perturbed text]. Intuitively, we may take the augmented [Perturbed logic, Perturbed text] to be not only the training example for the generator but also the positive sample for the evaluator, while crossover pairs [Seed logic, Perturbed text] and [Perturbed logic,

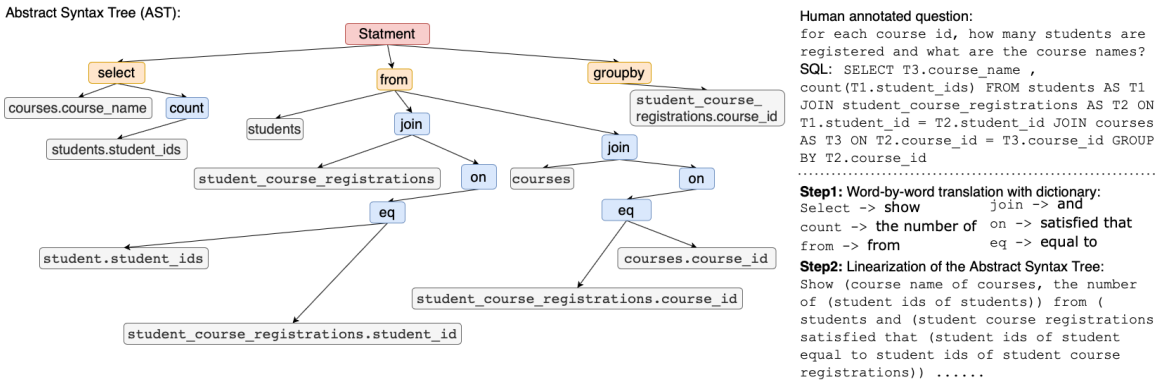

Figure 3: The example of word-by-word translation and linearization of structure-aware encoding.

Seed text] would be suitable negative samples for the evaluator.

3.3 Structure-aware Encoding

The logical forms normally have equivalent structured representations to precisely express the complex relations between a set of objects. For instance, the executable codes written in Python or SQL could be parsed into abstract syntax tree(AST) (Noonan, 1985) denoting the mutual relations among occurred constructs in the source code, while the knowledge bases may be converted into knowledge graphs that depict the relations between entities with directed edges. Compared to the plain text inputs, the structure-aware encoding capturing not only the sequential information from texts but also the internal logic from structural representations recently has been proved to be more effective in several Graph-to-Text tasks (Song et al., 2018; Ribeiro et al., 2020). To make full use of the intrinsic knowledge of the pre-trained BART model, we follow the similar approach proposed by Ribeiro et al. (2020) to linearize the structural representations of the SQL queries and logical forms respectively (Figure 3). Furthermore, the logical forms from different domains or datasets may vary in keywords, so normalizing them into a unified form would bridge the gaps between different logic NLG datasets and then increase the generalization ability of our framework. Hence, the logical forms would be firstly word-by-word translated into the unified intermediate semi-textual forms according to a manually annotated dictionary. Then the parenthesis is inserted into the semi-textual forms to denote the hierarchy of the correlated structured representations such as ASTs.

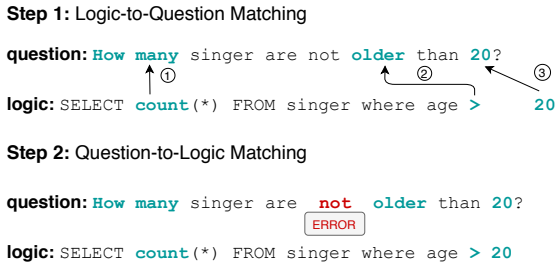

Figure 4: A sample of BLEC. The words marked in green are the matched tokens while the words marked in red are the tokens with no match.

4 BLEC for Logic Consistency Evaluation

Because the general-purpose automatic metrics such as BLEU, ROUGE, and BLEURT are not ideal for explicitly measuring the logic consistency, we propose BLEC, a new rule-based automatic evaluation metric called Bidirectional Logic Evaluation of Consistency. We apply a bidirectional evaluation to determine the logical consistency of pairs of logical forms and questions. The intuition behind this metric is that some key tokens such as number, operator, and keywords in the logical form should always be matched with some tokens that represent similar meanings in the question, and vice versa. An example is shown in Figure 4, BLEC first traverses the key tokens in the question, trying to find the tokens with the same meaning in the logic form to match them. Then, in step two, the sample is marked as inconsistent because there is one token with no match from the question to the logical form.

Formally, given a logical form $L = l_1, l_2, ..., l_n$ containing n word tokens and a questions $Q = q1, q2, ..., q_m$ containing m word tokens, the proposed evaluation metric performs token level

Dataset		Train	Dev	Test
SQL2Text	Generator	5600	1400	1034
	Evaluator	-	1142	1142
Logic2Text	Generator	8566	1095	1092
	Evaluator	-	1041	1041

Table 1: The statistics of the SQL2Text and Logic2Text dataset.

matching on l_i and q_j to test the consistency. To be specific, the matching procedure contains two steps, i.e. matching from L to Q as well as matching from Q to L. In step one, each key token l_i^{key} in L tries to match with the tokens in Q. In step two, each key token q_j^{key} in Q tries to match with the tokens in L. If no tokens are found that could be matched with any key tokens in either step one or step two, the sample will be marked as negative, vice versa. The final score is the accuracy of all the samples:

$$\text{BLEC} = \frac{\sum_{s \in S} match(s)}{|S|} \quad (2)$$

Where S denotes the dataset while $match(*)$ is the matching function with binary output, i.e. 1 for positive and 0 for negative.

Compared with the neural network evaluator requiring data-specific training, BLEC can be easily deployed to different datasets. In our experiments, we demonstrate that BLEC can be applied to two different datasets of text generation from two types of semantic parse input, and it shows a substantial agreement with human evaluation for evaluating logic consistency between the semantic parse input and the text output (Table 2).

5 Experiment Settings

5.1 Datasets

Text generation from semantic parses has different forms depending on the input formal representation. To demonstrate that our SNOWBALL and BLEC can be applied to different types of inputs, we study two tasks: (1) SQL2Text with the SQL query as the input and (2) Logic2Text with the logic forms as the input.

To this end, we make use of two existing publicly available datasets: For SQL2Text, we use the Spider dataset (Yu et al., 2018), a complex cross-domain semantic parsing and text-to-SQL dataset. Generating natural language from formal languages with abundant logic representations could be regarded as the inverse semantics parsing process. Therefore, we reverse the input and output as a dataset for the text generation from SQL queries with complicated logic. As the test set of the Spider dataset remains undisclosed, 20% of the original Spider training set is converted into a development set, and 80% of the training set remains to be the training set, and the original development set is exploited as the test set for our SQL2Text task. For Logic2Text, we use an existing Logic2Text dataset from Chen et al. (2020b). We pick the SENT and LOGIC_STR fields from the original Logic2Text to compose our own train data. We then change SENT to TEXTand change LOGIC_SET to LOGIC as our one keyword of each sample in the dictionary of our dataset.

In contrast, evaluating the logical consistency between logical form and text is closely related to the sequence classification tasks such as fact verification and natural language inference (NLI). According to the best of our knowledge, there is no existing dataset for evaluating the logical consistency between logical form and generated text. Therefore, we simplified the logic evaluation as a two-sequence binary classification problem and then construct the dataset with the development set and test set dedicated for our proposed evaluator. The dataset is constituted from the development and test set of Spider and Logic2text by three methods: (i) The [logical form, Text] pairs in the two datasets are regarded as positive samples; (ii) The human-labeled negative samples by intentionally introducing the logical inconsistency to the known [logical form, Text] pairs in the two datasets; (iii) The manually scored [logical form, Text] prediction given by the trained generator on the two datasets which contain both positive and negative samples. As for the human-labeled negative samples, we attempt to cover the possible logic perturbations mentioned in section 3.2 with minimum modification to the original [logical form, Text] pairs. For example, a coincident pair [`SELECT avg(age) FROM dogs`, What is the **average** age of dogs?] would be corrupted into [`SELECT avg(age) FROM dogs`, What is the **oldest** age of dogs?]. Table 1 summarizes the statistics of each dataset for both generator and evaluator, respectively.

5.2 Baselines and Implementation Details

The baselines for assessing the performance of SNOWBALL framework are the attention-based LSTM machine translation model (Tao et al., 2019), and the single-pass trained models which are the models trained before performing SNOWBALL iteration. For instance, the `BART-large` generator trained in the second SNOWBALL iteration would be compared to the identical `BART-large` generator in the zero SNOWBALL iteration. The hype-parameter settings of the models trained on *SQL2Text* and *Logic2Text*, mostly follow the default setting of BART model from Huggingface (Lewis et al., 2020; Wolf et al., 2020). However, the learning rate of evaluator and tokenizer are slightly different, namely the learning rate of evaluator on *SQL2Text* is 2e-5 for `BART-base` and is 5e-6 for `BART-large`, while the learning rate of evaluator on *Logic2Text* is 1e-5 for both `BART-base` and `BART-large`.

5.3 Multitask Learning

Due to the lack of data of logic NLG, intuitively collaborative training on *SQL2Text* and *Logic2Text* dataset may prevent the models from bias fitting to their confined training data. Aside from the standard special separators used by BART tokenizer, we further introduce *[SQL]* and *[logic]* tokens to be the control codes to indicate if one sample is from *SQL2Text* or *Logic2Text* dataset, similar as (Keskar et al., 2019). For each sample fed into the BART model, a corresponding control token is contacted in the front of the input logical form according to that sample source. Therefore, the distribution $p(Q_{SQL}|L_{SQL}, [SQL])$ of the *SQL2Text* models and $p(Q_{Logic}|L_{Logic}, [logic])$ of the *Logic2Text* models could be learned respectively during the backpropagation that takes the control tokens into account, while training the generator and evaluator in the MTL fashion.

5.4 Human Evaluation

To evaluate if the sentence generated by the model is logically consistent, we randomly sample 90 questions from the test set of Spider and a test set of logic2text separately to form a human evaluation set. The samples of each setting will be divided into two parts and assigned to two different annotators. Each part contains 10 overlap and 40 non-overlap examples, which means one person has to label 50 samples for a setting. As for the human evaluation, the annotators label the [logical form, text] as True or False based on two criteria: (1) the logic consistency between logical form and text; (2) The grammaticality of the text. After labeling, we estimate the accuracy of the model predictions by computing the expectation of the true labels from 80 non-overlap data. To prove the consistency of the annotators, we use the 10 overlap data to calculate the cohen kappa score. Only if the kappa score is over 0.4 which implies that this estimated accuracy is valid, the results would be reported. In Table 4, we only human annotated the results given by the models trained without snowball iteration and trained with 4 snowball iterations. It demonstrates the correlation between human evaluation and BLEC metrics in these two time steps instead of directly evaluating the improvement of model performance.

6 Results and Analysis

6.1 Correlation Analysis on BLEC

To show that BLEC is consistent with human judgment, we test the Pearson correlation between the BLEC score and the human evaluation result. We also include ROUGE and BLEU for comparisons. Therefore, we apply these four automatic metrics (BLEU, ROUGE, BLEURT, BLEC) to a human-labeled dataset and compare the evaluation results. This dataset is constructed by extracting 50 samples from each of the different Snowball iterations, 15 iterations in total. As shown in Table 2, the logic consistency between BLEC and human evaluation is 0.66 while BLEU, ROUGE, and BLEURT obtain scores below or around zero. This shows that the BLEC score is capable of testing the logical consistency between logic forms and questions.

Metrics	Pearson	p-value
ROUGE (Lin, 2004)	-0.30	0.28
BLEU (Papineni et al., 2002)	-0.24	0.40
BLEURT (Sellam et al., 2020)	+0.02	0.94
BLEC	**+0.66**	**0.007**

Table 2: The correlation with human annotation of BLEU, ROUGE, BLEURT, and our proposed BLEC.

6.2 Effectiveness of Snowball Framework

Generator The experimental results of the generator in our SNOWBALL framework are shown in Table 4. We found that the SNOWBALL training framework empirically leads to the improvement of

Input SQL:	SELECT count(*), max(Percentage) FROM country_language WHERE LANGUAGE = "Spanish" GROUP BY CountryCode
Pre-processed SQL:	(the number of (all items)) , (the maximum of (percentage)) that belongs to (countrylanguage) , that have ((language) equal to (spanish)) , grouped by (countrycode)
Label:	What is the total number of countries where **Spanish** is spoken by the **largest** percentage of people?
BART-base	the number and percentage of languages that are **Spanish** for each country code.
+snowball (iteration = 4)	the number of languages and **maximum** percentage of languages in each country code?
+multi-tasking	How many languages does **Spanish** have in each country code?
+snowball (iteration = 4)	Show the number and **maximum** percentage for each country code.
BART-large	How many languages are there in each country and what is the percentage of the language spoken in that country?
+snowball (iteration = 4)	Find the number of languages and **maximum** percentage of **Spanish** for each country code.
+multi-tasking	Find the number and percentage of speakers of **Spanish** in each country code.
+snowball (iteration = 4)	Find the number and **highest** percentage of speakers of **Spanish** for each country code

Table 3: Example outputs from different models with or w/o performing the MTL and SNOWBALL iteration.

	SQL2Text Test Set								
Metrics	BLEC						Human		
Snowball	-	1	2	3	4	5	-	4	κ
LSTM Seq2Seq	22.6								
BART-base	76.4	78.6	78.5	**84.1**	79.7	78.1	22	45	0.69
+MTL	89.1	**89.5**	89.2	88.9	88.6	88.1	66	68	0.5
BART-large	91.8	91.3	**93.7**	91.8	93.2	93.0	75	74	0.7
	Logic2Text Test Set								
Metrics	BLEC						Human		
Snowball	-	1	2	3	4	5	-	4	κ
LSTM Seq2Seq	41.1								
BART-base	87.9	86.1	**88.6**	87.4	87.7	87.8	83	85	0.48
BART-large	86.7	**87.8**	85.2	87.1	86.0	88.5	86	78	0.48

Table 4: The results of SNOWBALL generator using BLEC and human evaluation over different iterations.

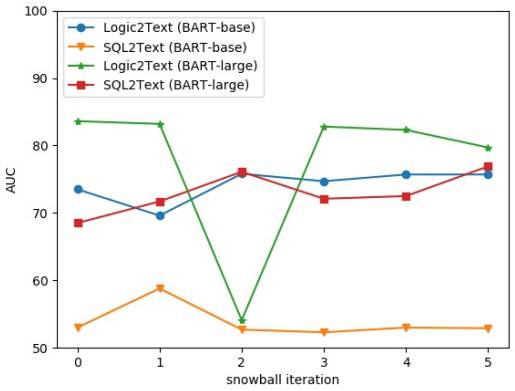

Figure 5: The result of SNOWBALL evaluator based on the AUC scores on the test set. The iteration = 0 means the model is under the regular training procedure as described in Figure 2.

the logic consistency. Evaluated by our proposed BLEC metric, the performance of BART-base generator improves the logic consistency by 10.1% on SQL2Text and by 0.7% on Logic2Text. Similarly, the performance of BART-large generator acquires the improvement by 2.1% on SQL2Text and by 1.2% on Logic2Text. Under the MTL setting on SQL2Text, the logic faithfulness of theBART-base generator is further enhanced by 16.6% compared to single-pass training, and is even boosted by 17.1% by combing with SNOWBALL training.

Evaluator The results of the evaluator in our SNOWBALL framework are illustrated in Figure 5. Empirically the snowball framework is more effective to the evaluators base on BART-base than BART-large, this is likely because that the BART-large models have already obtained enough intrinsic knowledge to accurately judge the validness of the [Logical form, Text] pairs. The data augmentation procedure of the SNOWBALL framework may introduce unexpected noise to the evaluators, which may cause a catastrophic reduction in terms of AUC and other metrics. On the other hand, the snowball framework indeed enhances the performance of the evaluator based on relative inferior BART-base by improving the performance on the *SQL2Text* by 10.9% as well as the *Logic2Text* by 3.1%. These results indicate that our proposed SNOWBALL framework is most suitable for tasks suffering from both domain data scarcity and the lack of external knowledge.

6.3 Case Study

Table 3 shows example outputs from our model with different settings. Apparently, in this case, the entity **Spanish** and the aggregator **maximum** are the touchstones for evaluating the logic consistency of each model. The prediction from the BART-large based generator trained under both snowball and multi-tasking frameworks simultaneously is the only one that acquires the seamless sentences from the input SQL. Furthermore, we also notice that multi-tasking learning significantly alleviates the artifacts within the generated text. Based on

the fact that, compared to the vanilla generators, the generator solely trained with snowball framework would enhance the logic consistency but also increase an unnatural sense to the generated sentences at the same time, we may argue that there is a trade-off between fluency and logic consistency of our purposed snowball framework. The model-level modification may collaboratively enhance the fluency and logic consistency of the NLG, which we would remain for future studies.

7 Conclusion

In this paper, we propose SNOWBALL, a neural network-based framework to augment the data alternatively by a generator, and an evaluator. In addition, we propose BLEC, an automatic evaluation metric that could evaluate the logic consistency between question and logic forms by directional matching. We also formulate two datasets and the experimental results show the effectiveness of the proposed framework. This method is applicable to other Data-to-Text tasks, because domain-specific rules for perturbations can be derived for most structural data with pre-defined structures or grammar.

8 Ethics Statement

The datasets we use are built by selecting and processing from two datasets that are open to the public, separately. The data sources we utilize to construct our datasets are Spider and Logic2Text, two complex and cross-domain text-to-SQL datasets. Besides, we use three experts to annotate about 500 data beyond the original dataset. We admit that some biases may still exist in our datasets, even though we have double-checked the data they annotated and the data from the original datasets.

Authors with SQL expertise annotate and verify our datasets through 1) selecting about 500 representative samples from the original dataset, 2) changing the entities of the samples, 3) using three different labels to mark which type of change has been done to the sentences, and 4) double-checking the quality of the data we annotate.

We conduct several experiments of different settings on our AWS server, with 8 Tesla V100 GPUs, to test the efficiency of our models. To be more specific, our experiments contain two different types. The first type of them is that we train both generator and evaluator using SQL2Text or Logic2Text. The second type of them is that we utilize SQL2Text and Logic2Text to train generators, use one of them to train evaluators in the first epoch, and train the next several epochs with both of them.

References

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Zhangming Chan, Xiuying Chen, Yongliang Wang, Juntao Li, Zhiqiang Zhang, Kun Gai, Dongyan Zhao, and Rui Yan. 2019. Stick to the facts: Learning towards a fidelity-oriented E-commerce product description generation. 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 4959–4968, Hong Kong, China. Association for Computational Linguistics.

Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. 2020a. Logical natural language generation from open-domain tables. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7929–7942, Online. Association for Computational Linguistics.

Zhiyu Chen, Wenhu Chen, Hanwen Zha, Xiyou Zhou, Yunkai Zhang, Sairam Sundaresan, and William Yang Wang. 2020b. Logic2Text: High-fidelity natural language generation from logical forms. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2096–2111, Online. Association for Computational Linguistics.

Marco Damonte and Shay B Cohen. 2019. Structural neural encoders for amr-to-text generation. *arXiv preprint arXiv:1903.11410*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Claire Gardent and Agnes Plainfossé. 1990. Generating from a deep structure. In *COLNG 1990 Volume 2: Papers presented to the 13th International Conference on Computational Linguistics*.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *J. Artif. Intell. Res.*, 61:65–170.

Hamza Harkous, Isabel Groves, and Amir Saffari. 2020. Have your text and use it too! end-to-end neural data-to-text generation with semantic fidelity. *CoRR*, abs/2004.06577.

Xing Hu, Ge Li, Xin Xia, David Lo, and Zhi Jin. 2020. Deep code comment generation with hybrid lexical and syntactical information. *Empir. Softw. Eng.*, 25(3):2179–2217.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. *arXiv preprint arXiv:1704.08760*.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A conditional transformer language model for controllable generation. *CoRR*.

Georgia Koutrika, Alkis Simitsis, and Yannis E. Ioannidis. 2010a. Explaining structured queries in natural language. In *Proceedings of the 26th International Conference on Data Engineering, ICDE 2010, March 1-6, 2010, Long Beach, California, USA*, pages 333–344.

Georgia Koutrika, Alkis Simitsis, and Yannis E Ioannidis. 2010b. Explaining structured queries in natural language. In *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, pages 333–344. IEEE.

Wojciech Kryscinski, Bryan McCann, Caiming Xiong, and Richard Socher. 2020. Evaluating the factual consistency of abstractive text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9332–9346, Online. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Abhijit Mishra, Anirban Laha, Karthik Sankaranarayanan, Parag Jain, and Saravanan Krishnan. 2019. Storytelling from structured data and knowledge graphs : An NLG perspective. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 43–48, Florence, Italy. Association for Computational Linguistics.

Axel-Cyrille Ngonga Ngomo, Lorenz Bühmann, Christina Unger, Jens Lehmann, and Daniel Gerber. 2013. Sorry, i don't speak SPARQL: translating SPARQL queries into natural language. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, pages 977–988.

Axel-Cyrille Ngonga Ngomo, Lorenz Bühmann, Christina Unger, Jens Lehmann, and Daniel Gerber. 2013. Sorry, i don't speak sparql: translating sparql queries into natural language. In *Proceedings of the 22nd international conference on World Wide Web*, pages 977–988.

Feng Nie, Jinpeng Wang, Jin-Ge Yao, Rong Pan, and Chin-Yew Lin. 2018. Operation-guided neural networks for high fidelity data-to-text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3879–3889, Brussels, Belgium. Association for Computational Linguistics.

Robert E. Noonan. 1985. An algorithm for generating abstract syntax trees. *Comput. Lang.*, 10(3/4):225–236.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017a. Why we need new evaluation metrics for NLG. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252, Copenhagen, Denmark. Association for Computational Linguistics.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017b. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, Saarbrücken, Germany. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186, Online. Association for Computational Linguistics.

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. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2020. Investigating pretrained language models for graph-to-text generation. *CoRR*, abs/2007.08426.

Leonardo FR Ribeiro, Claire Gardent, and Iryna Gurevych. 2019. Enhancing amr-to-text generation with dual graph representations. *arXiv preprint arXiv:1909.00352*.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for AMR-to-text generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1616–1626, Melbourne, Australia. Association for Computational Linguistics.

Yifeng Tao, Bruno Godefroy, Guillaume Genthial, and Christopher Potts. 2019. Effective feature representation for clinical text concept extraction. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 1–14, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Ran Tian, Shashi Narayan, Thibault Sellam, and Ankur P Parikh. 2019. Sticking to the facts: Confident decoding for faithful data-to-text generation. *arXiv preprint arXiv:1910.08684*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

Hongmin Wang. 2019. Revisiting challenges in data-to-text generation with fact grounding. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 311–322, Tokyo, Japan. Association for Computational Linguistics.

Qingyun Wang, Xiaoman Pan, Lifu Huang, Boliang Zhang, Zhiying Jiang, Heng Ji, and Kevin Knight. 2018. Describing a knowledge base. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 10–21, Tilburg University, The Netherlands. Association for Computational Linguistics.

Zhenyi Wang, Xiaoyang Wang, Bang An, Dong Yu, and Changyou Chen. 2020a. Towards faithful neural table-to-text generation with content-matching constraints. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1072–1086, Online. Association for Computational Linguistics.

Zhenyi Wang, Xiaoyang Wang, Bang An, Dong Yu, and Changyou Chen. 2020b. Towards faithful neural table-to-text generation with content-matching constraints.

Michael White. 2006. Efficient realization of coordinate structures in combinatory categorial grammar. *Research on Language and Computation*, 4(1):39–75.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, and Vadim Sheinin. 2018. Sql-to-text generation with graph-to-sequence model. *arXiv preprint arXiv:1809.05255*.

Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. Sqlizer: query synthesis from natural language. *Proceedings of the ACM on Programming Languages*, 1(OOPSLA):1–26.

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. 2019. 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 R. 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, Brussels, Belgium, October 31 - November 4, 2018, pages 3911–3921.

A BLEC Details

BLEC uses bidirectional keyword matching to detect the logic consistency. Table 5 shows several sample cases for constructing BLEC metrics. As shown in the table, the first column shows the type of matching rules. "Special" means that these rules are only contained in one of the datasets. Then, the following 3 columns display the tokens in different languages. Using the tokens, the algorithm can detect if the question matches the parse. For instance, given a pair of question and SQL parse, the algorithm could check if "MAX" is in SQL parse. If so, it will try to match one of the possible tokens corresponding to "MAX", i.e. largest/ greatest, etc., in the question, and vice versa. It is worth noting that, this table only shows a small part of the algorithm, however, all the rules can be classified as one of the three types.

Type	Spider	Logic2text	Natural Language
Negation	NOT	not_eq	not/ none...
Operator	$>$	filter_greater	larger/ more/ greater...
	$<$	filter_smaller	smaller/ less/ fewer...
	MAX	max	largest/ greatest...
	COUNT	count	total/ how many...
Special	ASC	-	ascending/ fewest...
	DESC	-	descending/ highest...
	-	most_str_eq	majority/ most...

Table 5: Sample rules for BLEC.