LOGEN: Few-Shot Logical Knowledge-Conditioned Text Generation With Self-Training

Shumin Deng, Member, IEEE, Jiacheng Yang, Hongbin Ye, Chuanqi Tan, Mosha Chen, Songfang Huang, Fei Huang, Huajun Chen, Member, IEEE, and Ningyu Zhang, Member, IEEE

Abstract—Natural language generation from structured data mainly focuses on surface-level descriptions, suffering from uncontrollable content selection and low fidelity. Previous works leverage logical forms to facilitate logical knowledge-conditioned text generation. Though achieving remarkable progress, they are data-hungry, which makes the adoption for real-world applications challenging with limited data. To this end, this paper proposes a unified framework for logical knowledge-conditioned text generation in the few-shot setting. With only a few seeds logical forms (e.g., 20/100 shot), our approach leverages self-training and samples pseudo logical forms based on content and structure consistency. Experimental results demonstrate that our approach can obtain better few-shot performance than baselines.

Index Terms—Few-shot, Self-training, Text Generation.

I. INTRODUCTION

Natural language generation (NLG) from structured data has good application prospects in communicating with humans in a natural way [1], such as financial report [2], medical report [3] and so on. However, previous studies [4] mostly concentrate on surface descriptions from simple records, such as limited schema (e.g., E2E [5], and WikiBio [6]), which suffer from low fidelity and uncontrollable content selection. Logical forms can condition the generation beyond superficial facts (e.g., “Canada has got 3 gold medals.”) with new statements that can be entailed from these facts (e.g., “Canada obtained the most gold medals”). [7] first leverage logical forms for NLG, and the generation module is provided with the table information and a logical form representing the target text’s semantics (see Fig. 1 for an example).

However, the success of this approach is heavily dependent on the availability of a massive amount of labeled training data, e.g., 10.8 k logic-text training pairs for the LOGIC2TEXT dataset [7] in a single domain. Such data-hungry nature makes NLG systems challenging to be widely adopted in real-world applications. To this end, we focus on exploring how to efficiently model for few-shot logical knowledge-conditioned text generation, which is not well-studied before.

To address the few-shot issue, one of the most potent methods is meta-learning, which transfers the experience learned from
similar tasks to the target task [8], [9], [10]. However, they have difficulty tackling text generation, mainly attributed to the excessive time cost required to train numerous instances. Another intriguing idea is to leverage unlabeled data via semi-supervised learning, which is useful for improving model performance when the target domain lacks manual resources. Self-training is a classical, intuitive and straightforward semi-supervised learning method, which first trains the model with labeled data and then enlarges the labeled set according to the most confident predictions (a.k.a., pseudo labels) on unlabeled data [11].

Note that there exists a quantity of in-domain raw text; it is intuitive to generate pseudo logical forms and leverage those unlabeled data via self-training. However, there are still several nontrivial challenges for self-training in text generation. Since there are only a few parallel data for training, those pseudo logical forms may contain many ill-posed samples, which may no longer bring performance improvement but even deteriorate the performance when noisy instances exceed the model’s robustness. Specifically, those pseudo logical forms should guarantee the content and structure consistency. Content consistency indicates that the generated logical forms have a consistent semantic meaning aligned with the input text. Structure consistency refers to the fact that the symbolic structure conforms to the logical specifications.

To alleviate the aforementioned problems, we propose a unified framework for the few-shot LOGical knowledge-conditioned text generation, namely LOGEN. Our approach utilizes self-training to leverage those easy-to-obtain in-domain corpora. Specifically, we utilize a sequence-to-sequence model (e.g., text-to-logic) trained with few-shot seed data to generate pseudo logical forms. To guarantee content consistency and structure consistency, we employ two key components. The first one is a content-consistency module. We utilize the reverse task of pseudo-logical form generation (i.e., target logical knowledge-conditioned text generation) to recover the input text. Our assumption states that, if the recovered text is semantically similar to the input text, the generated symbolic logical form will be of high quality. The second component is a structure-consistency module. We design a logical rationality estimator based on the general rules. Specifically, we convert the generated logical forms to trees and leverage domain rules to calculate logical rationality. Finally, we obtain the quality score of each generated logical form. We then select the top-K instances as high-quality logical forms and train the model iteratively until no unlabeled data remain.

In summary, our main contributions include:

- We study the few-shot logical knowledge-conditioned text generation problem, which is a new branch of research that has not been well-explored to the best of our knowledge.
- We propose the LOGEN framework, which leverages self-training and samples pseudo-logical forms based on content and structure consistency.
- Experimental results on the benchmark dataset illustrate that our approach can achieve better performance than baselines in the few-shot setting.

II. RELATED WORK

NLG from structured data has been appealed to researchers for many years [12], [13], [14], [15], [16], [17], resulting in many real-world applications, including those for the automatic generation of weather reports [18], sport reports [19], and clinical reports [20], [21]. Previous approaches typically utilize pipeline-based approaches that included surface realization and content selection [22], [23]. More recent models tend to leverage end-to-end neural network for tasks such as table-to-text generation [19], [24], [25], [26]. AMR-to-text generation [27], [28], [29], [30], graph-to-text generation [31], [32], [33], and so on. Though achieving good performance on surface-level NLG, they still suffer from low fidelity and uncontrollable content selection [34].

To address this issue, it is intuitive to leverage external logical knowledge for better generation [35], [36], [37], [38], [39], [34] firstly proposes text generation using logical inferences from a table. Their study mainly supports probing purposes or evaluates neural models’ ability to generate logically correct descriptions based solely on the table content. Note that the best model in [34] only achieves better than 20% factual correctness rate according to a follow-on human evaluation. Thus, the formulation of this approach still misses the mark for real-world text generation systems due to the low fidelity and uncontrollability. Text2Logic [7] formulates NLG as a logical form to the text generation problem. Alongside the table information, the model is provided with the logical form. However, its performance relies on the availability of large numbers of supervised data (i.e., logic–text pairs), thus restricting its applicability.

Our work relates to the few-shot NLG. TableGPT [4] focuses on generating high-fidelity text for the table-to-text generation using limited training pairs. Another work [40] propose a few-shot NLG approach with language modeling to compose coherent sentences with content selection. However, those approaches are trained and tested mainly on surface-level descriptions, which are not straightforwardly applicable to the logical knowledge-conditioned text generation. From a methodological perspective, our work relies on self-training [41], [42] which has shown some surprising success with natural language processing (NLP) tasks [43], [44], [45], [46]. Our work also relates to dual learning [47], [48] which tackle the training data bottleneck through a dual-learning game. Differently, we integrate the dual tasks into the self-training framework.

III. METHODOLOGY

A. Problem Definition

The goal of logical knowledge-conditioned text generation is to generate natural language $Y$ from tables $T$ conditioned on logical forms $L$. Given an input table $t_i \in T$ with a logical form $l_i \in L$ as a condition, we follow [7] to linearize the table content $t_i$ and the logical form $l_i$ and then concatenate them to obtain the input sequence $x_i$. In the few-shot setting, we have an extremely small parallel dataset with $T = \{x_i, y_i\}_i^M$ and many unlabeled texts $U = \{u_j\}_j^N$, where $M \gg N$. Note that
it is easy to obtain a large scale of unlabeled and diverse text corpus, but rather difficult to acquire their corresponding logical forms. Our target is the mapping function, \( Logic_{2}Text \), between \( x_i \) and \( y_i \). Formally, we have:

\[
y_i = Logic_{2}Text(x_i, \phi),
\]

where \( x_i \) is the concatenation of input logical form and table content (including table captions and headers), \( \phi \) is the parameter of \( Logic_{2}Text \), and \( y_i \) is the output text.

### B. Framework

As shown in Fig. 2, we regard logical knowledge-conditioned text generation as a sequence-to-sequence task and introduce the encoder and decoder architecture in Section III-C and self-training in Section III-D. To select high-quality samples from pseudo-logical forms, we introduce the content consistency module, which leverages the reverse task of \( Text_{2}Logic \) to estimate the semantic consistency score (\( text \to logic \to text' \)) in Section III-D1. Furthermore, we introduce a structure consistency module with rules to score those instances in Section II-D2. Finally, we introduce the overall optimization procedure and training details in Section III-E.

### C. Encoder–Decoder for Logical Knowledge-Conditioned Text Generation

We utilize the pre-trained language model as an encoder. Specifically, we leverage the generative pre-trained transformer GPT-2 [49] as the backbone, following [40]. Note that our approach is model-agnostic, and other architectures, such as UniLM [50], and BART [51], can be applied. We concatenate the table content \( t \) and logical forms \( l \) as input sentences following [7]. We leverage the transformer to encode each sentence as vectors. We utilize the same architecture with different parameters for the pseudo logical form generation (i.e., \( Text_{2}Logic \) generation). Because lots of output sequences share the same tokens with the nodes in the input logical trees, we introduce a logic-tree-based copy mechanism for decoding.

**Logic-tree-based Copy Mechanism:** We first leverage a gate that decouples the framework into language-model-based generation and tree-node selection [52]. We leverage a soft gate, \( p_{copy} \), to choose between copying from logic-tree nodes using attention weights as the probability distribution or generating from softmax-over-vocabulary:

\[
p_{copy} = \sigma(W_c e_t + W_s s_t + W_x x_t + b),
\]
where \( x_t, s_t \) are the decoder input, state, respectively. \( \sigma \) refers to the sigmoid activation function. \( c_t = \sum a_t^i h_t^i \) and \( h_t \) is the encoder hidden state at time step \( t \). \( W_c, W_a, W_x \), and \( b \) are trainable parameters. We optimize the copy probability, \( p_{\text{copy}} \), using an additional loss as follows:

\[
L = L_c + \lambda \sum_{w_j \in V_i} (1 - p_{\text{copy}}^j),
\]

where \( L_c \) is the cross-entropy loss (original loss) between the model outputs and target texts, \( \{ V_i \} \) is the input logic-tree-node list, \( w_j \) is the target token at position \( j \), and \( \lambda \) is a hyperparameter of the weight for this copy-loss term.

### D. Self-Training

In a vanilla self-training framework, a tagger is first initialized using a set of instances having gold labels. Then, the tagger is used to tag a set of unlabeled data, and the tagging confidence for each unlabeled instance is evaluated. The automatically labeled instances having the highest confidence is added to the training set using the labels predicted by the tagger. Correspondingly, the unlabeled instance is removed from the unlabeled dataset. The tagger is then retrained using the updated training dataset and is used to tag and select the unlabeled instances from the remaining dataset. In this paper, we leverage the Text2Logic model as the tagger because 1) the unannotated texts related to tables are diverse and easy to obtain; 2) the quality of logical forms is easier to control; 3) the simplicity of applying to existing data-to-text datasets with only a few annotated logical forms. In the next section, we introduce the quality-control strategy of content and structure consistency.

1) **Content Consistency**: Inspired by back translation [53], [54], [55], we leverage the reverse task of Text2Logic to estimate the semantic consistency score. Back translation is proposed for machine translations wherein the sentence in the source language (e.g., Chinese) would be translated to the target language (e.g., English) and would then be translated back to the source language (e.g., Chinese). The essence of semantic consistency is that a variable, \( x \), and a bijective mapping function, \( f() \), should satisfy \( \hat{x} = f^{-1}(f(x)) \), where \( f^{-1} \) is the inverse function of \( f \). Formally, given the pre-trained Text2Logic and Logic2Text models and text sample \( u \in U \), we have

\[
\hat{x} = \text{Text2Logic}(u),
\]

\[
u' = \text{Logic2Text}(\hat{x}).
\]

Specifically, given the original text \( u \), and the recovered text \( u' \), we obtain the semantic consistency score as:

\[
\text{score}_{\text{content}} = \frac{(1 + \beta^2) \cdot R_{\text{ucs}} \cdot P_{\text{ucs}}}{R_{\text{ucs}} + \beta^2 \cdot P_{\text{ucs}}},
\]

where \( R_{\text{ucs}} \) and \( P_{\text{ucs}} \) refer to the longest common subsequence regarding \( u \) and \( u' \), respectively. Specifically, we obtain \( R_{\text{ucs}} = \frac{\text{LCS}(u,u')}{\text{len}(u')} \) via calculating the longest common subsequence regarding \( u \) and \( u' \), and obtain \( P_{\text{ucs}} = \frac{\text{LCS}(u,u')}{\text{len}(u)} \) via calculating the longest common subsequence regarding \( u' \) and \( u \). \( \beta \) is a hyper-parameter, and we utilize a development set to tune optimized \( \beta \). Because logical forms are a specific data form that differs from raw text, they should obey some structure constraints. For example, the function in the generated logical form should appear in the pre-defined schema, and the number of parameters (i.e., child nodes in the logic tree) should follow the function definition. Thus, we introduce a structure-consistency module to score these instances.

2) **Structure Consistency**: For a generated logical form \( L \), we design several general rules to estimate the logical rationality score. Note that these rules are orthogonal to different types of logical forms, such as \( \lambda \)-calculus and Prolog. Thus, they can be easily applied to other datasets.

**Rule1: Logic Consistency**. Given the generated logical form \( L \), if the parentheses do not match, then the rule does not hold.

**Rule2: Function Mutual Exclusion**. Given the generated logical form \( L \), with the entire function set \( O \), and the default function set \( F \) (e.g., argmax, sum). The default operation set is defined based on the schema, if \( \exists o \in O \) and \( o \notin F \), then the rule does not hold.

**Rule3: Parameter Consistency**. Given the generated logical form \( L \), with the entire function set \( O \), and the default function set \( F \), if \( \exists o \in O \) and \( \text{nodes}_{O}(o) \neq \text{nodes}_{F}(o) \), the logical rationality score is zero. Otherwise, the score is one. \( \text{nodes}_{O} \) and \( \text{nodes}_{F} \) denote the number of parameters of the generated and default function \( o \). We calculate the average parameter consistency for all nodes in the logic tree using the breadth-first search. If the average score is lower than \( \kappa \), then the rule does not hold.

### E. Training Details

After obtaining the content consistency and structure consistency scores, we utilize an instance-sampling approach to select the top-\( K \) instances. The overall algorithm is shown in Algorithm 1.

### IV. EXPERIMENT

#### A. Dataset and Metric

We evaluate our approach on the benchmark dataset, LOGIC2TEXT [34]. We employ seven types of the most commonly used logics [34]: count, superlative, comparative, aggregation, majority, unique, and ordinal. The LOGIC2TEXT dataset contains 7,566,1,000, and 1,095 samples for training, validation, and testing, respectively. The maximum length of a natural-language segment in the dataset is 130 words, and more than 90% of the data items are less than 90 words in length. The overall statistics of the LOGIC2TEXT dataset are shown in Table II. For automatic evaluations, we employ BLEU-1,2 ROUGE-1, 2, and L (F-measure),3 noted as B-1, R-1, R-2, and R-L.

---

1We have experiment with Logic2Text with automatically produced logical forms, but obtain little improvement.

2Standard script NIST mteval-v13a.pl.

3rouge-1.5.5.
Algorithm 1: Self-training with Content and Structure Consistency.

Require: train set \( T = \{ (x_i, y_i) \}_{i=1}^N \) with text and logic pairs, text corpus \( U = \{ y_j \}_{j=1}^M \), \( \lambda, \kappa \)

Require: random shuffle \( U \)

1: while \( U \) is not empty do
2: \( \text{Text2Logic}() \leftarrow \text{train}(Y, X) \)
3: \( \text{Logic2Text}() \leftarrow \text{train}(X, Y) \)
4: \( X \leftarrow \text{tag}(\text{Text2Logic}(U)) \)
5: \( U \leftarrow \text{tag}(\text{Logic2Text}(X)) \)
6: Calculate \( \text{score}_{\text{content}} \) with (6)
7: Sample top \( K \) instances \( \hat{u} \) with \( \text{score}_{\text{content}} \) obeying Rule 1,2, and 3 from \( U \)
8: \( T \leftarrow T \cup \{ \hat{x}, \hat{u} \} \)
9: \( U \leftarrow U \setminus \{ \hat{x}, \hat{u} \} \)
10: Return \( \text{Logic2Text}() \)

B. Setting

We utilize GPT-2 (124 M) as the backbone from [49]. We also utilize BART-base [51] as the backbone to further verify the effectiveness of the framework. We employ Adam [56] as the optimizer, the initial learning rate \( \alpha \) is set to 2e-5. \( K \) is set to 1,000, \( \kappa \) is set to 0.5, and the batch size is 32. We tune the hyperparameters on the development set. We train the model on eight NVIDIA V100 16 GB GPU, with the patience of 4 epochs for each iteration. We set a maximum number of the loop (with early stopping) in the algorithm to avoid an infinite loop. We run each experiment five times and calculate average performance.

C. Baseline

TableGPT [4]: We employ the TableGPT which leverages table structure reconstruction and content matching without logical forms for few-shot table-to-text generation.

Seq2seq+att: We employ the seq2seq+att with the attention model following [57]. We concatenate the table cation, header, and the logical form as the input sequence.

Transformer+copy: Following [7], we leverage the Transformer structure with the copy mechanism as a baseline.

BART [51]: We employ BART-base which is a denoising autoencoder for pretraining Seq2Seq models.

GPT-2: We leverage the generative pre-training model GPT-2 (124 M) from [49].

We evaluate our framework LOGEN in the few-shot setting with only a few training instances (20/100/500 shots via random sampling) and regard all the other data in the training set as unannotated data.

D. Main Results

From Table I, we observe that GPT-2 achieves better performance in all few-shot settings, as also observed by [7]. We also observe that TableGPT obtains poor performance and even fails to compete with Transformer+copy, which illustrates the advantage of logic guidance. We notice that our LOGEN yields better performance with 23.57% for F1 22.45% of Rouge-L by only 20 golden labels, which demonstrates the effectiveness of our approach. Note that the R-L score of fully-supervised setting with GPT-2 is 53.04 (We have reproduced this score.) [7], our model obtain even better performance with only 500-shot instances (6% of the dataset). We also notice that our approach with BART-base as the backbone can yield better performance than baselines, further verifying the framework’s effectiveness. We also utilize BLEU-4 for evaluation and notice that our approach with 100-shot instances obtains a comparative performance of 16.95 than GPT-2 with 17.06, illustrating the limitations of LOGEN. We think this may cause by the few-shot data, leading to unstable performance.

E. Ablation Study

We conduct an ablation study to validate the effectiveness of the different components. \( w/o \) content and \( w/o \) structure refer to a model lacking content and structure consistency, respectively. \( w/o \) logic-copy refers to the model without logic-tree-based copy. From Fig. 4, we observe that all models have a performance decay without content/structure consistency and logic-copy, indicating that all components are beneficial. We also notice that the content consistency is sensitive to the Rouge score, revealing that content consistency may be more important.

F. Human Evaluation Results

We conduct a human evaluation to evaluate the generated answer summaries from three aspects: (1) Informativity: How well does the text capture the key information from the original table? (2) Logicalness: How logically is the text correlated to the input logical form and table content? and (3) Readability: How fluent and coherent is the text? We randomly sample 100 instances and generate their output text using four methods (i.e., TableGPT, Seq2seq+att, GPT-2, and LOGEN) and variations of our approach (\( w/o \) all refers to the vanilla self-training approach with GPT-2). Three data annotators with a Ph.D. degree are asked to score each generated text on a scale of 1 to 5 (higher is better). They are firstly trained with 100 sentences for evaluation to well understand the three metrics of informativity, logicalness and readability. We then ask them to annotate sampled instances to evaluate whether the there annotator could label the three metrics correctly. We do this three times to ensure the annotator can indeed make good and consistent decisions. We then ask the three annotators to evaluate the generated instances. Due to the time and budget limit, we follow [7] to sample 200 examples from each method for evaluation. We also calculate the average inter-rater agreement between annotators using Fleiss’ kappa scores [58], finding that five of six annotations showed good agreement (\( \kappa = 0.9 \)).
Table I

| # Training instances | 20     | 100    | 500    |
|----------------------|--------|--------|--------|
| Metrics              | B-1    | R-1    | R-2    | R-L    | B-1    | R-1    | R-2    | R-L    | B-1    | R-1    | R-2    | R-L    |
| TableGPT             | 14.29  | 16.25  | 2.54   | 15.31  | 23.02  | 24.61  | 4.33   | 21.58  | 27.52  | 28.23  | 6.67   | 25.15  |
| Seq2Seq+att          | 13.31  | 13.59  | 2.39   | 14.54  | 23.87  | 25.13  | 3.67   | 21.33  | 31.13  | 33.16  | 10.35  | 30.33  |
| Transformer+copy     | 15.35  | 16.87  | 3.56   | 15.87  | 26.98  | 27.35  | 5.77   | 23.25  | 33.51  | 35.15  | 12.35  | 32.45  |
| BART                 | 23.51  | 24.12  | 13.35  | 20.14  | 37.32  | 39.31  | 16.71  | 33.31  | 43.03  | 42.17  | 20.35  | 37.25  |
| GPT-2                | 23.75  | 24.13  | 14.91  | 21.19  | 47.33  | 48.16  | 23.60  | 38.54  | 54.89  | 55.56  | 29.82  | 45.60  |

LOGEN (BART)         | 42.13  | 45.02  | 20.14  | 40.02  | 49.34  | 51.30  | 29.18  | 49.44  | 51.32  | 52.89  | 32.19  | 51.53  |
LOGEN (GPT-2)        | 47.32  | 49.03  | 24.35  | 43.64  | 56.33  | 57.23  | 31.10  | 51.10  | 57.32  | 59.05  | 34.08  | 53.18  |

Table II

| General Statistics of LOGIC2Text |
|----------------------------------|
| Tables                           | 5,554 |
| Examples                         | 10,753 |
| Vocabulary                       | 14.0k |
| Avg. description length          | 16.77 |
| Avg. # nodes in logical form     | 9.00  |
| Avg. # function nodes in logical form | 3.27  |
| Avg. length of the linearized logical form | 24.35 |

LOGEN Output:

mexico had the 2nd highest total in athletics at the 1935 central american and caribbean games.

w/o logic Output:

cuba had the 1st highest total in athletics at the 1935 central american and caribbean games.

mexico -> puerto:
puerto had the 2nd highest total in athletics at the 1935 central american and caribbean games.

nth_argmax -> nth_argmin:
mexico had the 2nd lowest total in athletics at the 1935 central american and caribbean games.

Table III

Human Evaluation Results

| Models      | Info | Logic | Read |
|-------------|------|-------|------|
| TableGPT    | 2.22 | 2.20  | 3.01 |
| Seq2Seq+att | 2.33 | 2.15  | 3.13 |
| BART        | 2.88 | 2.35  | 3.92 |
| GPT-2       | 2.78 | 2.45  | 3.88 |

LOGEN       | 3.98 | 4.54  | 4.35 |

w/o content | 3.62 | 4.31  | 3.93 |
w/o structure | 3.52 | 4.21  | 3.95 |
w/o all     | 3.02 | 2.67  | 3.90 |

Table III lists the human evaluation results, showing that our approach consistently outperforms the other methods in all aspects. We observe that TableGPT (without logic) achieves the lowest logic score. Note that TableGPT-generated text lacks an explicit logical form. Thus, the model has trouble generating logical correct text. Logical forms significantly affect the logic performance scores. Seq2Seq+att generate text using a sequence-to-sequence model, resulting in the low-quality text in the few-shot setting. GPT-2 and BART achieve relatively low scores in informativity and logic, which may be caused by the failure of NLG in the few-shot setting. However, GPT-2 and BART generate more fluent text having higher readability scores, which may have taken advantage of the pre-trained language model. w/o content and w/o structure obtain a performance drop compared with LOGEN, further indicating the effectiveness of the different components. w/o all obtains only a small
Fig. 5. Error analysis with different logic types.

performance improvement in the human evaluation compared with GPT-2, revealing that the quality of the self-labeled logical form influences model performance.

G. Manipulating Text With Logic

To analyze the effect of logical forms for text generation, we randomly sample from the instance and conduct an experiment. From Fig. 3, we observe that, without logic condition, the model misses some important entities or logic types (argmax) and are logically wrong. We further notice that, when we permute the logical forms with different entities (Sep 21 to Sep 15) or functions (argmax to argmin), our model generate corresponding text with the logic condition, which indicates that logical forms can guide text generation, thus, promoting the logical correctness of NLG.

H. Analysis

Error Analysis: We conduct an error analysis of our approach. From Fig. 5, we observe that text generation with the logic type of comparative obtain the most deficient performance, indicating that the model still suffers from numerical logic reasoning. Moreover, we observe that the results of all logic types are still far from satisfactory. This indicates that few-shot generation is rather challenging and may require additional schemes and extra information to improve.

Impact of Different $K$: Furthermore, we investigate the performance with different $K$ regarding the number of instances in each iteration. From Fig. 6, we observe that, with an increase in the number of training steps, the model gradually achieves better performance. We also notice that the model obtain comparable performance when $K$ is 500 or 1,000. Because a small $K$ leads to more iterations that require more computing resources, we set $K = 1,000$ to balance performance and computation complexity.

Impact of Different Instances: Finally, we study the problem of which samples to choose at each iteration to promote future works. We divide the LOGIC2TEXT dataset into easy, middle, and hard subsets based on the logical trees’ layer depths. Specifically, we regard the brackets, { and }, as layer dividers for the logical form. Intuitively, a sample having a large layer depth should be more complex and difficult to predict. We obtain 1,943 hard instances, 4,068 middle instances, and 2,555 easy instances. Fig. 7 illustrates the samples chosen for each iteration by LOGEN. The green, yellow, and red bars refer to the easy, middle, and hard instances. We notice that, during the early stage of training, the model is prone to choosing those easy instances, whereas, during the last stage, the model mostly chooses the difficult samples. We think this is because, during the early stage, the model can not obtain high qualified pseudo instances with only few-shot training samples. When the iteration increases, the model performance increases and more hard instances can be tagged with qualified generation targets (pseudo data). This observation indicates that our model implicitly learns the training curriculum for self-training.

V. CONCLUSION

This article studies the few-shot logical knowledge-conditioned text generation problem and proposes a unified framework, LOGEN. Experimental results indicate that our approach achieves better performance than baselines on the benchmark dataset. With our approach, we successfully generate text with logic guidance using only a few seeded training instances,
which can be applied to many real-world data-to-text generation applications. Our framework is general in the sense that any generation model with different logical types can be employed. In the future, we plan to study the problem of controlled NLG without logical forms (i.e., zero-shot logical NLG) and to extend our approach to more challenging tasks in which logical forms cannot be induced using a tree-style.

V. BROADER IMPACT STATEMENT

A broad goal of NLG is to generate fully-synthetic, faithfully representative text segments to facilitate data sharing. For example, it is of high value in the medical domain and provides a social benefit to generate emergency department-chief complaints, a history of present illness, or the progress notes from electronic health records. However, previous large-scale pre-trained language model (e.g., GPT-2/3) still lack the ability to generate logical correct texts, thus, missing the mark for real-world text generation system. Our approach can leverage only a few logical forms to generate fidelity and logically correct descriptions of these reports, promoting the fulfillment of NLG applications. Our vision is to develop a logical controllable text generation system for the NLP community, and our innovation is a small step in that direction. Our framework may fail when integrated with illegal or malicious logical forms, thus, generating unintended texts. We leave this for future works.

ACKNOWLEDGMENT

We want to express gratitude to the anonymous reviewers for their hard work and kind comments. We thank Ning Ding for helpful discussions and feedback on this paper.

REFERENCES

[1] N. Zhang et al., “Alicg: Fine-grained and evaluable conceptual graph construction for semantic search at alibaba,” in Proc. 27th ACM SIGKDD Conf. Knowl. Discov. Data Mining, 2021, pp. 3893–3905, doi: 10.1145/3447548.3467057.

[2] S. Murakami et al., “Learning to generate market comments from stock prices,” in Proc. 55th Ann. Meeting Assoc. Comput. Linguistics, 2017, pp. 1374–1384, doi: 10.18653/v1/P17-1126.

[3] S. A. Hasan and O. Farri, “Clinical natural language processing with deep learning,” in Data Science for Healthcare. Berlin, Germany: Springer, 2019, pp. 147–171.[Online]. Available: https://pubmed.ncbi.nlm.nih.gov/31794016/

[4] H. Gong et al., “Tablegpt: Few-shot table-to-text generation with table structure reconstruction and content matching,” in Proc. 29th Int. Conf. Comput. Linguistics, 2020, pp. 1978–1988, doi: 10.18653/v1/2020.coling-main.179.

[5] O. Dusek, J. Novikova, and V. Rieser, “Evaluating the state-of-the-art of end-to-end natural language generation: The E2E NLG challenge,” Comput. Speech Lang., vol. 59, pp. 123–156, 2020, doi: 10.1016/j.csl.2019.06.009.

[6] R. Lebret, D. Grangier, and M. Auli, “Neural text generation from structured data with application to the biography domain,” in Proc. Conf. Empirical Methods Natural Lang. Process., 2016, pp. 1203–1213.[Online]. Available: http://aclweb.org/anthology/D16/D16-1128.pdf

[7] Z. Chen et al., “Logic2text: High-fidelity natural language generation from logical forms,” in Proc. Conf. Empirical Methods Natural Lang. Process., 2020, pp. 2096–2111, doi: 10.18653/v1/2020.findings-emnlp.190.

[8] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in Proc. 34th Int. Conf. Mach. Learn., 2017, vol. 70, pp. 1126–1135.[Online]. Available: http://proceedings.mlr.press/v70/finn17a.html

[9] N. Zhang et al., “Long-tail relation extraction via knowledge embeddings and graph convolution networks,” in Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol., 2019, pp. 3016–3025, doi: 10.18653/v1/n19-1306.

[10] N. Zhang, S. Deng, Z. Sun, J. Chen, W. Zhang, and H. Chen, “Relation adversarial network for low resource knowledge graph completion,” in Proc. Web Conf., 2020, pp. 1–12, doi: 10.18653/v1/2020.acl-main.224.

[11] A. Oliver, A. Odena, C. Rauff, E. D. Cubuk, and I. J. Goodfellow, “Relativistic evaluation of deep semi-supervised learning algorithms,” in Proc. Adv. Neural Inf. Process. Syst.: Annu. Conf. Neural Inf. Process. Syst., 2018, pp. 3239–3250.[Online]. Available: https://proceedings.neurips.cc/paper/2018/hash/c1fca270c84e807908d7d06d26bab52-A Vertx.html

[12] C. Zhao, M. A. Walker, and S. Chaturvedi, “Bringing the structural gap between encoding and decoding for data-to-text generation,” in Proc. 58th Ann. Meeting Assoc. Comput. Linguistics, 2020, pp. 2481–2491, doi: 10.18653/v1/2020.acl-main.224.

[13] Z. Wang, X. Wang, B. An, D. Yu, and C. Chen, “Towards faithful neural table-to-text generation with content-matching constraints,” in Proc. 58th Ann. Meeting Assoc. Comput. Linguistics, 2020, pp. 1072–1086, doi: 10.18653/v1/2020.acl-main.101.

[14] X. Shen, E. Chang, H. Su, C. Niu, and D. Klakow, “Neural data-to-text generation via jointly learning the segmentation and correspondence,” in Proc. 58th Ann. Meeting Assoc. Comput. Linguistics, 2020, pp. 7155–7165.[Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.641

[15] E. Chang, J. Caplinger, A. Marin, X. Shen, and V. Dembarg, “DART: A lightweight quality-suggestive data-to-text annotation tool,” in Proc. 28th Int. Conf. Comput. Linguistics: Syst. Demonstrations, 2020, pp. 12–17.[Online]. Available: https://www.aclweb.org/anthology/2020.coling-demos.3

[16] W. Chen, Y. Su, X. Yan, and W. Y. Wang, “KGPT: Knowledge-grounded pre-training for data-to-text generation,” in Proc. Conf. Empirical Methods Natural Lang. Process., 2020, pp. 8635–8648.[Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-main.497

[17] H. Shahidi, M. Li, and J. Lin, “Two birds, one stone: A simple, unified model for text generation from structured and unstructured data,” in Proc. 58th Ann. Meeting Assoc. Comput. Linguistics, 2020, pp. 3864–3870.[Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.355

[18] P. Liang, M. I. Jordan, and D. Klein, “Learning semantic correspondences with less supervision,” in Proc. 47th Ann. Meeting Assoc. Comput. Linguistics 4th Int. Joint Conf. Natural Lang. Process., 2009, pp. 91–99.[Online]. Available: http://www.aclweb.org/anthology/P09-1011

[19] S. Wiseman, S. M. Shieber, and A. M. Rush, “Challenges in data-to-document generation,” in Proc. Conf. Empirical Methods Natural Lang. Process., 2017, pp. 2253–2263.[Online]. Available: https://aclanthology.info/papers/D17-1239/D17-1239

[20] C. DiMarco et al., “The development of a natural language generation system for personalized e-health information,” in Proc. Medinfo: 12th World Congr. Health (Med.) Inform.; Buildin Sustainable Health Syst., 2007, Art. no. 2339.[Online]. Available: http://www.cs.cmu.edu/~hovy/papers/07Medinfo-healthdoc.pdf

[21] S. H. Lee, “Natural language generation for electronic health records,” NPJ Digit. Med., vol. 1, no. 1, 2018, Art. no. 63.[Online]. Available: https://www.nature.com/articles/s41746-018-0070-0

[22] E. Reiter and R. Dale, “Building applied natural language generation systems,” Natural Lang. Eng., vol. 3, no. 1, pp. 57–87, 1997, doi: 10.1017/S1351324997001502.

[23] A. Gatti and E. Krahmer, “Survey of the state of the art in natural language generation: Core tasks, applications and evaluation,” J. Artif. Intell. Res., vol. 61, pp. 65–170, 2018, doi: 10.1613/jair.5477.

[24] T. Liu, K. Wang, L. Sha, B. Chang, and Z. Sui, “Table-to-text generation by structure-aware seq2seq learning,” in Proc. 32nd AAAI Conf. Artif. Intell., 30th Innov. Appl. Artif. Intell., 8th AAAI Symp. Educ. Adv. Artif. Intell., 2018, pp. 4881–4888.[Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16789

[25] H. Gong, X. Feng, B. Qin, and T. Liu, “Table-to-text generation with effective hierarchical encoder on three dimensions (row, column and time),” in Proc. Conf. Empirical Methods Natural Lang. Process., 9th Int. Joint Conf. Natural Lang. Process., 2019, pp. 3141–3150, doi: 10.18653/v1/D19-1310.

[26] A. P. Parikh et al., “ToTTo: A controlled table-to-text generation dataset,” in Proc. 2020 Conf. Empirical Methods Natural Lang. Process., Nov. 16–20, 2020, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds., Association for Computational Linguistics, pp. 1173–1186.[Online]. Available: https://doi.org/10.18653/v1/2020.emnlp-main.89
[27] L. Song, Y. Zhang, Z. Wang, and D. Gildea, “A graph-to-sequence model for AMR-to-text generation,” in Proc. 56th Annu. Meeting Assoc. Comput. Linguistics, 2018, pp. 1616–1626. [Online]. Available: https://www.aclweb.org/anthology/P18-1150

[28] M. Dattolo and S. B. Cohen, “Structural neural encoders for AMR-to-text generation,” in Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol., 2019, pp. 3649–3658. [Online]. Available: https://www.aclweb.org/anthology/N19-1366

[29] Y. Zhao, L. Chen, Z. Chen, R. Cao, S. Zhu, and K. Yu, “Line graph enhanced AMR-to-text generation with mix-order graph attention networks,” in Proc. 58th Annu. Meeting Assoc. Comput. Linguistics, 2020, pp. 732–741. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.67

[30] S. Yao, T. Wang, and X. Wan, “Heterogeneous graph transformer for graph-to-sequence learning,” in Proc. 58th Annu. Meeting Assoc. Comput. Linguistics, 2020, pp. 7145–7154. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.640

[31] C. Zhao, M. Walker, and S. Chaturvedi, “Bringing the structural gap between encoding and decoding for data-to-text generation,” in Proc. 58th Annu. Meeting Assoc. Comput. Linguistics, 2020, pp. 2481–2491. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.224

[32] L. Song et al., “Structural information preserving for graph-to-text generation,” in Proc. 58th Annu. Meeting Assoc. Comput. Linguistics, 2020, pp. 7987–7989. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.712

[33] W. Chen, J. Y. Chen, Z. S. Chen, and W. Y. Wang, “Logical natural language generation from open-domain tables,” in Proc. 58th Annu. Meeting Assoc. Comput. Linguistics, 2020, pp. 7929–7942, doi: 10.18653/v1/2020.acl-main.708.

[34] N. Zhang, S. Deng, J. Li, X. Chen, W. Zhang, and H. Chen, “Summarizing chinese medical answer with graph convolution networks and question-focused dual attention,” in Proc. Findings Assoc. Comput. Linguistics: EMNLP, 2020, pp. 15–24, doi: 10.18653/v1/2020.findings-emnlp.2.

[35] X. Chen et al., “Knowprompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction,” in Proc. ACM Web Conf., F. Laforet, R. Troncy, E. Simperl, D. Agarwal, A. Gionis, I. Herman, and L. Médini, Eds., Virtual Event, Lyon, France: ACM, Apr. 25-29, 2022, pp. 2778–2788. [Online]. Available: https://doi.org/10.1145/3485447.3511998

[36] S. Deng et al., “Ontoed: Low-resource event detection with ontology embedding,” in Proc. 59th Annu. Meeting Assoc. Comput. Linguistics, 11th Int. Joint Conf. Natural Lang. Process., 2021, pp. 2828–2839, doi: 10.18653/v1/2021.acl-long.220.

[37] C. Li et al., “Sentiprompt: Sentiment knowledge enhanced prompt-tuning for aspect-based sentiment analysis,” 2021, arXiv:2109.08306.

[38] S. Deng et al., “Low-resource extension with knowledge-aware pairwise prototype learning,” Knowl.-Based Syst., vol. 235, 2022, Art. no. 107584. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705121008467

[39] Z. Chen, H. Eavani, W. Chen, Y. Liu, and W. Y. Wang, “Few-shot NLG with pre-trained language model,” in Proc. 58th Annu. Meeting Assoc. Comput. Linguistics, 2020, pp. 183–190, doi: 10.18653/v1/2020.acl-main.18.

[40] I. Triguero, S. García, and F. Herrera, “Self-labeled techniques for semi-supervised learning: Taxonomy, software and empirical study,” Knowl. Inf. Syst., vol. 42, no. 2, pp. 245–284, 2015, doi: 10.1007/s10115-013-0706-y.

[41] X. Li et al., “Learning to self-train for semi-supervised few-shot classification,” in Proc. Neural Inf. Process. Syst. Proc. Assoc. Comput. Inf. Process. Syst., 2019, pp. 10276–10286. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/bfb25365df26ae038f1a3a59c26687e80-Abstract.html

[42] S. Mukherjee and A. H. Awadallah, “Uncertainty-aware self-training for text classification with few labels,” 2020, arXiv:2006.15135.

[43] Y. Qing et al., “Text classification using label names only: A language model self-training approach,” in Proc. Conf. Empirical Methods Natural Lang. Process., 2020, pp. 9006–9017, doi: 10.18653/v1/2020.emnlp-main.724.

[44] J. Du et al., “Self-training improves pre-training for natural language understanding,” 2020, arXiv:2010.02194.

[45] Z. Qi et al., “Unsupervised knowledge graph alignment by probabilistic reasoning and semantic embedding,” in Proc. 30th Int. Joint Conf. Artif. Intell., 2021, pp. 2019–2025, doi: 10.24963/ijcai.2021/278.
Hongbin Ye is currently working toward the Ph.D. degree with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China. His research interests include information extraction, knowledge graph, and controllable neural natural language generation.

Chuanqi Tan received the Ph.D. degree from the School of Computer Science and Engineering, Beihang University, Beijing, China, in June 2019. He is currently an Algorithm Expert of Language Technology Lab, Alibaba DAMO Academy. His research interests include question answering, information extraction, and biomedical natural language processing.

Mosha Chen received the master’s degree from the Department of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai, China, in 2011. He is currently a Staff Algorithm Engineer with Alibaba Group. His research interests include natural language processing and medical AI technology.

Songfang Huang is currently a Senior Staff Algorithm Engineer with Language Technologies Lab, Alibaba DAMO Academy. He leads a team working on large-scale pre-trained language models and AI for Healthcare. Before joining Alibaba, he is also a Research Manager and Research Staff Member of IBM Research. His research interests include language modeling, question answering, information extraction, and cognitive healthcare.

Fei Huang is currently a Principal Researcher with Language Technologies Lab, Alibaba DAMO Academy. He leads R&D on NLP foundational technologies, dialogue, and machine translation. His team develops various NLP technologies ranging from lexical, syntactical, semantic, discourse, and deep learning-based algorithms, and integrate them into the Alibaba NLP platform, which supports several hundred internal and external clients with advanced NLP models, systems, and solutions in various industries.

Huajun Chen (Member, IEEE) received the bachelor’s and the Ph.D. degrees from Zhejiang University, Hangzhou, China, in 2000 and 2004, respectively. He is currently a Full Professor with the College of Computer Science and Technologies, Zhejiang University, Hangzhou, China, and the Director of the Joint Lab on Knowledge Engine of AZFT (Alibaba-Zhejiang University Joint Research Institute of Frontier Technologies), and Deputy Director of the Key Lab of Big Data Intelligence, Zhejiang Province. He was a Visiting Assistant Professor with the Yale Center for Medical Informatics, Yale University, New Haven, CT, USA, (From June 2006 to June 2007), and a Visiting Scholar with the School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, (From June 2007 to August 2008).

Ningyu Zhang (Member, IEEE) is currently an Associate Professor/Doctoral supervisor with Zhejiang University, Hangzhou, China, leading the group about KG and NLP technologies. He has authored or coauthored many papers in top international academic conferences and journals, such as Natural Machine Intelligence, Nature Communications, NeurIPS, ICLR, AAAI, IJCAI, WWW, KDD, SIGIR, ACL, ENNLP, NAACL, and IEEE/ACM Transactions on Audio Speech and Language. His research interests include natural language processing, information extraction, and large language models. He was the Area Chair of ACL 2023, ARR Action Editor, a Senior Program Committee Member of IJCAI 2023, Program Committee Member of AAAI, NeurIPS, ICLR, WWW, SIGIR, KDD, ICML, AAAI, and Reviewer of TKDE and TKDD.