Logical Natural Language Generation from Open-Domain Tables

Wenhu Chen[1], **Jianshu Chen**[2], **Yu Su**[3], **Zhiyu Chen**[1] **and William Yang Wang**[1]
University of California, Santa Barbara, CA, USA[1]
Tencent AI Lab, Bellevue, WA, USA[2]
The Ohio State University, Columbus, Ohio, USA[3]
{wenhuchen, zhiyuchen, william}@cs.ucsb.edu
jianshuchen@tencent.com, su.809@osu.edu

Abstract

Neural natural language generation (NLG) models have recently shown remarkable progress in fluency and coherence. However, existing studies on neural NLG are primarily focused on surface-level realizations with limited emphasis on logical inference, an important aspect of human thinking and language. In this paper, we suggest a new NLG task where a model is tasked with generating natural language statements that can be *logically entailed* by the facts in an open-domain semi-structured table. To facilitate the study of the proposed logical NLG problem, we use the existing TabFact dataset (Chen et al., 2019) featured with a wide range of logical/symbolic inferences as our testbed, and propose new automatic metrics to evaluate the fidelity of generation models w.r.t. logical inference. The new task poses challenges to the existing monotonic generation frameworks due to the mismatch between sequence order and logical order. In our experiments, we comprehensively survey different generation architectures (LSTM, Transformer, Pre-Trained LM) trained with different algorithms (RL, Adversarial Training, Coarse-to-Fine) on the dataset and made following observations: 1) Pre-Trained LM can significantly boost both the fluency and logical fidelity metrics, 2) RL and Adversarial Training are trading fluency for fidelity, 3) Coarse-to-Fine generation can help partially alleviate the fidelity issue while maintaining high language fluency. The code and data are available at https://github.com/wenhuchen/LogicNLG.

1 Introduction

Neural network models, especially the recent wave of massive models like BERT (Devlin et al., 2019) and GPT-2 (Radford et al., 2019), have shown the ability to generate natural language text at an astonishing level of fluency and coherence. For the generated text to fulfill its purpose, however, a critical property that is necessary but often overlooked is *fidelity*, i.e., what is generated should be faithful to the underlying data, knowledge, or meaning representation. A line of recent work has started to address the surface-level fidelity issue of natural language generation (NLG) by encouraging the model to learn to reuse the verbatim of certain inputs through copy mechanism (See et al., 2017; Gu et al., 2016; Wiseman et al., 2017; Liu et al., 2018), structured attention (Liu et al., 2018), or planning and selection/entity modeling (Puduppully et al., 2019a,b). While shown to be effective, most such methods so far are primarily focused on surface-level realization and simply restate the facts in the underlying data (Figure 1).

Medal Table from Tournament

Nation	Gold Medal	Silver Medal	Bronze Medal	Sports
Canada	3	1	2	Ice Hockey
Mexico	2	3	1	Baseball
Colombia	1	3	0	Roller Skating

Surface-level Generation

Sentence: Canada has got 3 gold medals in the tournament.
Sentence: Mexico got 3 silver medals and 1 bronze medal.

Logical Natural Language Generation

Sentence: Canada obtained 1 more gold medal than Mexico.
Sentence: Canada obtained the most gold medals in the game.

Figure 1: Table-to-text generation examples with and without implicit logical inference. Logical NLG requires a generation model to generate natural language statements that can be logically entailed by the facts in the table instead of simply restating certain superficial facts in natural language.

However, humans have the ability to generalize beyond superficial facts (e.g., "*Canada has got 3 gold medals.*") by inferring and communicating with new statements that can be entailed from these facts (e.g., "*Canada obtained the most gold medals.*"). We believe it is important for NLG models to be able to generalize beyond the superficla facts given to them as well. Therefore, we propose a new task, *logical NLG*, where a model is tasked

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7929–7942
July 5 - 10, 2020.

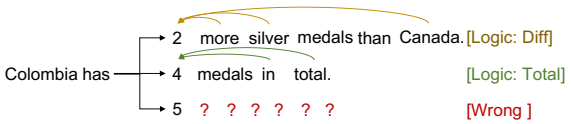

Figure 2: When making the decision at the third step, the model needs to foresee the future tokens to ensure logical consistency. There is no back-tracking once the model makes a wrong decision like "5".

with generating natural language statements that can be *logically entailed* by the given data (i.e., the *premises*). The new task requires a model to jointly reason and generate sentences that are consistent both linguistically and logically. Since there are a variety of reasoning/inference tasks such as natural language inference (Bowman et al., 2015) and commonsense reasoning (Talmor et al., 2019), to avoid confusion, this paper is specifically focused on inferences involving symbolic operations over the given table (Pasupat and Liang, 2015).

To empower research in this direction, we collect a new corpus LOGICNLG based on the existing TabFact (Chen et al., 2019), which brings two major renovations to the existing NLG paradigm: 1) the text involves diversified types of logical inferences including math operations like max/min/sum/add, comparison operations like same/different, and counting operations like total/only. A more detailed description of logical inference is listed in the Appendix. 2) while existing datasets are often restricted to a specific domain such as weather (Liang et al., 2009), restaurant (Dušek et al., 2019), NBA (Wiseman et al., 2017), etc, LOGICNLG uses open-domain tables without prior knowledge about their schema. As such, existing methods based on surface-level copying (See et al., 2017; Gu et al., 2016; Puduppully et al., 2019a) becomes insufficient, so are the existing fidelity evaluation based on the surface-level information extraction (Wiseman et al., 2017; Rohrbach et al., 2018; Dhingra et al., 2019), which extracts surface triples in a certain pre-defined form (i.e. subj-pred-obj, n-gram) and compare them with the surface content given in the knowledge.

Most neural generation models follow a monotonic generation schema from left to right with the current prediction only depending on the preceding words. Logical NLG poses unique challenges to the traditional generation scheme due to the mismatch between *sequence order* and *logical order*. As illustrated in Figure 2, the word "2" is derived from the logical inference of 'diff(Silver medal of Colombia, Silver medal of Canada)) $\rightarrow$ 2.' In other words, the logical order of word "2" should be after "more", "silver", and "Canada", while the sequence order of "2" is before those words. Since the monotonic generation scheme is purely based on sequence order while agnostic to logical order, existing NLG models struggle to maintain the fidelity as they cannot model the logical dependency on future tokens. To alleviate such an order mismatch, an NLG model must have the capability to plan ahead for the next few steps before generation. In this context, we believe LOGICNLG to be an important testbed to study such a planing/inference ability in generation models (Ford et al., 2018; Welleck et al., 2019). In this paper, we further propose a non-monotonic coarse-to-fine generation model and show that it is able to alleviate the order mismatch problem and achieve better performance. The contribution of this work is three-fold:

i) We propose a new research problem of logical natural language generation, and provide novel metrics to approximately evaluate the logical fidelity of generation models.

ii) We justify the mismatch problem between sequence order and logical order of the traditional monotonic generation scheme in logical NLG.

iii) We conduct comprehensive experiments with state-of-the-art neural generation models under both automatic and human evaluation, which demonstrates the challenges and opportunities for future research on logic NLG.

2 Dataset and Problem Definition

Existing NLG datasets (Chen and Mooney, 2008; Dušek et al., 2019; Lebret et al., 2016; Liang et al., 2009) are mainly composed of surface-level description over the given records. Though ROTOWIRE (Wiseman et al., 2017) involves sporadic inference in the long document, and the inference is restricted to domain-specific knowledge (e.g. double-double, smash, triple-double and other NBA-related terms). Hence, we need a better testbed for studying the proposed problem.

Statistics We construct a dataset based on TabFact (Chen et al., 2019), which is a table-based fact-checking dataset with rich logical inferences in the annotated statements. Specifically, we took their positive statements (the sentences which are en-

	Vocab	Examples	Vocab/Sent	Tables	Domain	Source	Inference	Schema
WEATHERGOV	394	22.1K	0.01	22.1K	Weather	Crawled	No	Known
WikiBIO	400K	728K	0.54	728K	Biography	Crawled	No	Limited
ROTOWIRE	11.3K	4.9K	0.72	4.9K	NBA	Annotated	Few	Known
LOGICNLG	122K	37.0K	**3.31**	7.3K	**Open**	Annotated	**Rich**	**Unlimited**

Table 1: Comparison of LOGICNLG against existing NLG datasets in different aspects.

Figure 3: Evaluation of surface-level generation vs. logical natural language generation. It suffices to use IE-based evaluation (Wiseman et al., 2017; Rohrbach et al., 2018) to verify surface-level generation, but it causes either "empty triple" or "false negative" problems to verify logical NLG.

tailed by the knowledge in the table) collected from "complex channel" (required to annotate sentences with logical inference) as our target text. To prevent confusion with the original dataset, we name this table-to-text dataset as LOGICNLG, which contains 28,450 training, 4,260 validation and 4,305 test examples based on 7,392 open-domain tables crawled from Wikipedia. Each table has 5 different examples covering diverse types of logical inference. More detailed statistics and comparisons are listed in Table 1. LOGICNLG is distinguished from the existing datasets due to:

i) It involves very rich logical inference, every annotated sentence involves certain types of inference with minimum domain-specific knowledge. The open-domain characteristic simulates a realistic setting, where we cannot enumerate the possible inference based on the scheme, which poses great challenges to the model's generalization capability.

ii) It is mainly composed of short sentences with an average length of 11 and a simple syntactic structure, which isolates from other linguistic complexity to focus on the problem of logical inference.

The dataset contains tables with open schema crawled from diversified domains Figure 4. The major categories are sports, politics, and entertainment. The schema diversity of the tables make the rule-based system infeasible to apply. Besides, most of the tables have very rich numeral records, which provide a great testbed for logical inference.

Problem Definition Here, we formally define our proposed table-to-text generation task. The input is a table $\mathbf{T}$ with its title denoted as a natural language sequence W. The table $\mathbf{T} = \{T_{i,j}|i \leq R_T, j \leq C_T\}$ has R_T rows and C_T columns with

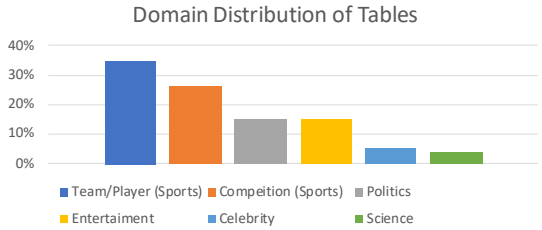

Figure 4: The domain distribution of LOGICNLG.

the T_{ij} being the content in the (i, j)-th cell. T_{ij} could be a word, a number, a phrase or even a natural language sentence. The annotated statement is a sentence $Y = y_1, y_2, \cdots, y_n$, we aim to train a neural generation model $p(Y|\mathbf{T})$ to generate statement $\hat{Y}$ which are both fluent and logically (numerically) supported by the given table $\mathbf{T}$.

3 Automatic Evaluation

In this section, we discuss the evaluation of our proposed NLG task. The fluency evaluation is simply based on the standard metrics like Perplexity (Bengio et al., 2003) and BLEU-1,2,3 (Papineni et al., 2002) based on NLTK (Bird, 2006). The most challenging problem is to evaluate the *logical fidelity* of the generated sentences, which is also the core problem of our paper. The existing IE-based extractive evaluation (Wiseman et al., 2017) leads to two issues as shown in Figure 3: 1) Empty Extraction: the sentence can not be formulated as (subject, predicate, object) structure, thus the IE system fail to extract triples for verification. 2) False Negative: the sentence is a logical composition (instead of surface form) of the fact from the table, the IE system cannot match it against the table. For these reasons, we test two approximate automatic metrics:

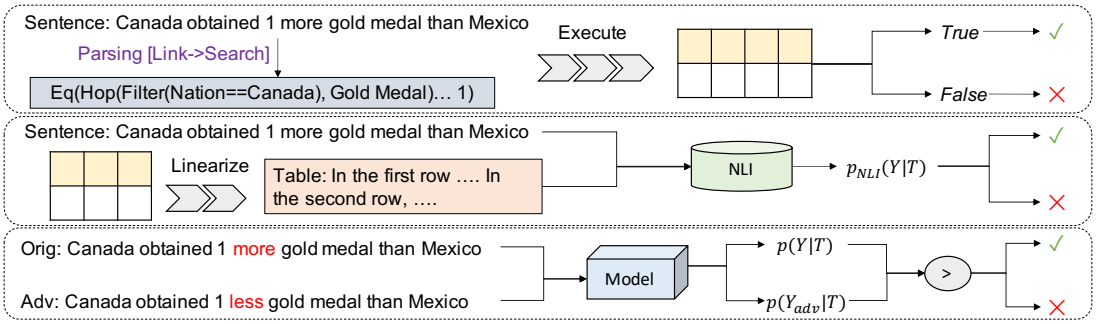

Figure 5: The parsing-based and adversarial evaluation to measure model's correctness in logical reasoning.

Parsing-based Evaluation We first propose a model-based evaluation method, which aims to directly extract the meaning representation from the generated sentence and execute it against the table to verify its correctness. Our evaluation is based on weakly-supervised semantic parsing (Liang et al., 2009, 2013), the basic idea is to first link entities and predicates in the sentence, and then use linked entities to perform a breadth-first search to synthesize potential logical forms, finally, a scorer is used to re-rank these logical forms and filter out spurious ones. The logical form returns a binary value of `True` to indicate whether its logic is supported by the knowledge. The basic idea is shown in the upper part of Figure 5, the implementation details are in the Appendix. We pre-train the semantic parser f_γ on the training set $(\mathbf{T}, Y) \in D_{train}$ with weakly supervised algorithm, at test time, we use it to parse a sentence Y into a set of logical forms, which is re-ranked to obtain the highest logical form P_{best}. We compute the ratio of P_{best} returning "true" on D_{test} to approximate model's fidelity.

$$\text{SP-Acc} = \mathop{\mathbb{E}}_{(\mathbf{T},\hat{Y})\in D_{test}} \mathbb{I}(P_{best} \rightarrow True | P_{best} = f_\gamma(\hat{Y}))$$

where $\mathbb{I}$ is the indicator function.

NLI-based Evaluation We then propose another model-based evaluation method to complement the parsing-based evaluation (which is sensitive to semantic variation), the basic idea follows (Kryściński et al., 2019) to evaluate the entailment score between the table and the generated sentence. The NLI model is based on Table-BERT (Chen et al., 2019), which linearizes the table into textual form and uses it as the evidence for natural language inference. The model is trained with TabFact (Chen et al., 2019) dataset containing both positive/negative samples. During the evaluation, we use this NLI model to predict the entailment relationship based on the likelihood of $p_{NLI}(Y|T)$. Finally, we compute the ratio of "entailed" to approximate model's fidelity:

$$\text{NLI-Acc} = \mathop{\mathbb{E}}_{(\mathbf{T},\hat{Y})\in D_{test}} \mathbb{I}(p_{NLI}(Y|\mathbf{T}) > 0.5)$$

where $\mathbb{I}$ is the indicator function.

Adversarial Evaluation Adversarial evaluation (Goodfellow et al., 2014; Kannan and Vinyals, 2017) is used to study the generation model's robustness in logical reasoning. Specifically, we hire human workers from Amazon Mechanical Turk[1] to annotate adversarial examples for the test/validation set by simply changing minimum words to revert the logic of the sentence. Such adversarial examples preserve linguistic components like length and style except the logic-related words to specifically disentangle the generation model's reasoning skill. As drawn in the lower part of Figure 5, the original sentence modifies its word "more" into "less" as an adversarial example. There are two principles the workers need to follow to make their jobs accepted: 1) the modified words/phrases should be roughly equally frequent to balance the language prior, for example, the number "1" is better swapped with "2,3" rather than "9999" which rarely appears in the corpus. 2) the perturbation should be diverse enough to cover different aspects of logical reasoning skills. We use the generation model $p(Y|\mathbf{T};\beta)$ to score the original sentence Y and the adversarial sentence Y_{adv}. If the confidence of the original example is higher than its adversarial counterpart, we count it as a successful defense, otherwise as a failed defense. We use the success rate to approximate model's logical reasoning capability.

$$\text{Adv-Acc} = \mathop{\mathbb{E}}_{(\mathbf{T},Y,Y_{adv})\in D_{test}} [\mathbb{I}(p(Y|\mathbf{T}) > p(Y_{adv}|\mathbf{T}))]$$

where $\mathbb{I}$ is the indicator function.

[1] https://www.mturk.com/

Discussion Both types of metrics have pros and cons, the SP-Acc and NLI-Acc are two metrics unbiased as it measures the peak samples in the model's likelihood, however, both metrics are based on imperfect models and thus their evaluation scores are inaccurate. SP-Acc is more sensitive to number/calculation errors, and NLI-Acc is more sensitive to semantic errors, therefore, we report both of them to help increase the metrics' robustness. In contrast, the adversarial evaluation score is accurate in terms of reflecting the model's reasoning capability on the given samples. However, as the provided samples might not lie in the high-confidence area of the model's distribution, it is biased in reflecting the model's general reasoning capability. Though these fidelity metric models are prone to errors, in section 6, we show their consistency with human judgment, which reveals their potential to assist human evaluation.

4 Baselines

In this section, we design comprehensive baseline models to perform logical NLG. Specifically, we consider the following two cases: non-pretrained models (LSTM/Transformer) with copy mechanism and pre-trained models (GPT-2 and BERT) with sub-word unit. We train these models with three different algorithms: Maximum Likelihood, Adversarial Training, and Reinforcement Learning.

4.1 Non-pretrained Models

Here we mainly consider two table encoding methods, namely field-infusing and field-gating. These two methods differ in their strategies to coalesce the field information into cells. After the table is represented as a sequence of vectors, a decoder based on LSTM (Hochreiter and Schmidhuber, 1997) or Transformer (Vaswani et al., 2017) is applied to generate text token by token. The two methods are depicted in the upper part of Figure 6:

Field-Infusing This strategy is inspired by Lebret et al. (2016). We first use an LSTM (Hochreiter and Schmidhuber, 1997) to encode the table field text word by word and then use the last output $\mathbf{z_i}$ as field representation. This representation is concatenated with the embedding of row index $\#j$ and word embedding at each cell to obtain a position-aware cell embedding $\mathbf{e_k}$ for each word inside the cell. We stack transformers layers on top of the cell embedding to obtain the table representation as $\mathbf{h_i} \in \mathbb{R}^D$ with D as the dimension.

Field-Gating This strategy is inspired by by Liu et al. (2018). Like the previous strategy, we first use an LSTM (Hochreiter and Schmidhuber, 1997) to obtain field representation $\mathbf{z_i}$. The field representation is concatenated with ending distance information as the input to an additional field gate built inside the LSTM as suggested in Liu et al. (2018), such a field gate is used to control whether the current cell is already encoded. Such a mechanism can help LSTM to identify the boundary between different cells to grasp local information.

4.2 Pre-trained Models

To further enhance the fluency and resolve the out-of-vocabulary problem, we use pre-trained language models and finetune them on LOGICNLG. Specifically, we consider two models based on GPT-2 (Radford et al., 2019) and BERT (Devlin et al., 2019), respectively, and name them as GPT-TableGen and BERT-TableGen.

Table Linearization We follow previous work on linearizing knowledge base as natural language (Liu et al., 2019; Zhang et al., 2019) to propose "table linearization", which uses template to flatten the table $\mathbf{T}$ as a document $P_T = w_1, \cdots, w_{|T|}$ fed into pre-trained language models to generate statement Y, where we use w_i to denote the i-th word in the generated paragraph P_T and $|T|$ to denote the length of the paragraph (the word w_i is either a table entry or a functional word in the template). As depicted in the left bottom part of Figure 6, the original table $\mathbf{T}$ is transformed into a paragraph by horizontally scanning each cell $T_{11} \rightarrow T_{1,C_T} \rightarrow T_{R_T,C_T}$ in the table.

GPT-TabGen we directly feed the paragraph P_T as the input to the pre-trained GPT-2 model and generate the output sentence Y. We finetune the model on LOGICNLG by maximizing the likelihood of $p(Y|P_T;\beta)$, with β denoting the parameters of GPT-2 model (Radford et al., 2019).

BERT-TabGen 1) we encode the linearized paragraph P_T using the pre-trained BERT model into the source representation $\mathbf{h_1}, \cdots, \mathbf{h_{|T|}}$. 2) at the i-th time step, we replace all the words in the groundtruth statement Y after i-th time step by <MASK> token and use BERT to encode the partially masked Y^i as $\mathbf{g_1^i}, \cdots, \mathbf{g_n^i}$. 3) we use an attention layer f_θ to obtain the output hidden states $\hat{\mathbf{g}}_\mathbf{1}^\mathbf{i}, \cdots, \hat{\mathbf{g}}_\mathbf{n}^\mathbf{i}$, where $\hat{\mathbf{g}}_\mathbf{i}^\mathbf{i}$ is used to predict the word $\hat{y}_i$. We jointly optimize β of BERT and θ to maximize

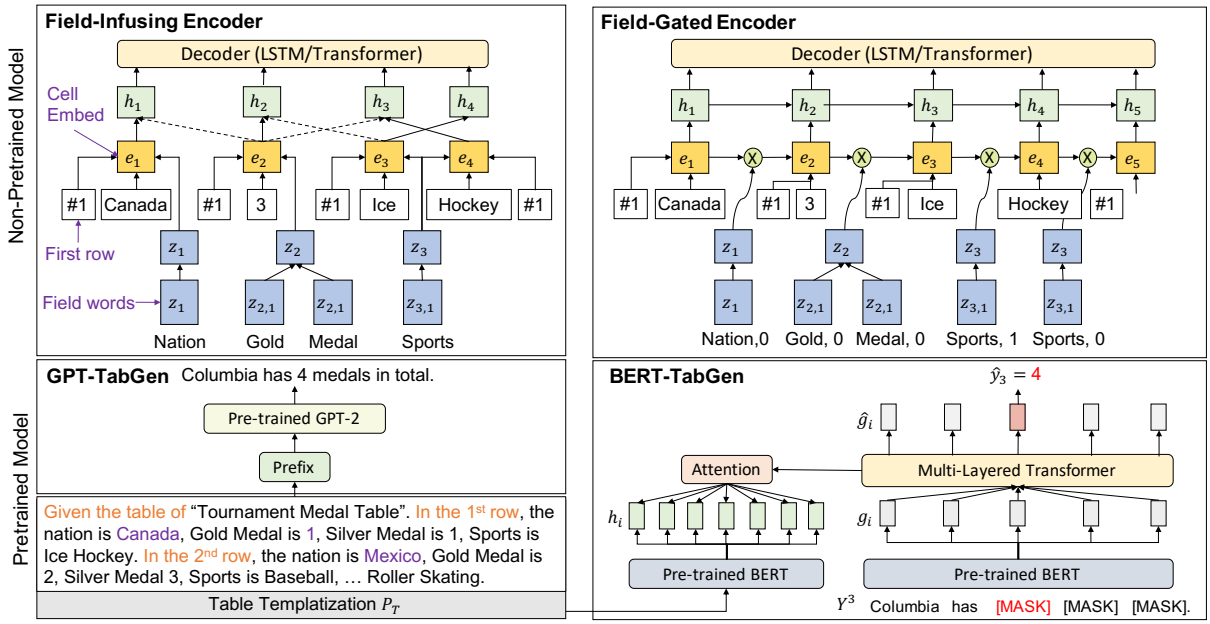

Figure 6: The Non-pretrained and Pre-trained generation models, the detailed table is shown in Figure 1.

the likelihood of generating text Y conditioned on the table and the masked partial sentence. As BERT is a bidirectional model, we need to re-encode the target sentence at each step to get $\mathbf{g^i_{1:n}}$. Therefore, the generation is finished with n passes.

4.3 Training

Except for the standard maximum likelihood training, we also use the following training algorithms:

Adversarial Regularization To encourage the model to ground on the table rather than relying on artificial language priors (Ramakrishnan et al., 2018), we use an adversarial regularization to enhance the maximum likelihood training. Specifically, we first perform entity resolution to locate all the numbers, count, entities in the sentence and then randomly replace them with entities or numbers appearing in the table $\mathbf{T}$. These perturbed samples Y_{adv} are used as adversarial examples to regularize the model's behavior. Formally, we optimize β to maximize the objective:

$$\operatorname*{argmax}_{\beta} \log p(Y|\mathbf{T};\beta) - \lambda \log p(Y_{adv}|\mathbf{T};\beta)$$

where λ is the controlling hyper-parameter.

Reinforcement Learning The maximum likelihood training is a fluency-driven objective, which is inconsistent with the goal of logical consistency. To bridge the gap, we view the generation problem from the reinforcement learning perspective to optimize the long-term fidelity. We use the trained semantic parser to assign reward to the policy $p(y_i|y_{1:i-1};\beta)$. At i-th step, the generator will sample different actions y_i and roll-out from $i+1$-th step to produce a full sequence starting from y_i using greedy search. The full sentence receives a binary score $r(Y,\mathbf{T})$ from the semantic parser as reward. Formally, we optimize the objective:

$$\operatorname*{argmax}_{\beta} \mathop{\mathbb{E}}_{y_i \sim p(y_i|y_{1:i-1})}[\mathop{\mathbb{E}}_{y_{i+1:n}}[r(y_{1:n},\mathbf{T})]] \log p(y_i|y_{1:i-1};\beta)$$

where we only use one trajectory to approximate the inner roll-out expectation for efficiency.

5 Coarse-to-Fine Generation

As discussed before, the baseline models follow the monotonic generation scheme and suffer from the mismatch between sequence order and logical order (Figure 2). In this section, we propose an imperfect remedy for such a situation based on the coarse-to-fine generation paradigm.

Before plunging into technical details, it is helpful to first realize the resemblance between logical NLG and semantic parsing (Dong and Lapata, 2018). Compared to traditional NLG tasks like machine translation and summarization, logical NLG is closer to semantic parsing in the sense that a model may make catastrophic errors that are impossible to be corrected at later steps (Figure 2). Therefore, we take inspiration from semantic parsing models (Dong and Lapata, 2018) that have proven effective in mitigating such errors and propose a coarse-to-fine generation scheme. We break down generation into two phases. In the first phase,

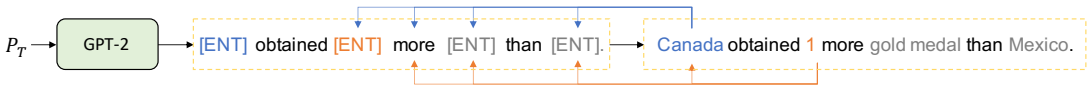

Figure 7: Coarse-to-fine generation scheme: first generates a template, and then realize the surface form. It exposes more context to the surface realization model for better capturing logical dependency.

the model only generates a template which determines the global logical structure, while in the second phase the model generates the final, grounded sentence conditioned on the template generated in the first phase. As depicted in Figure 7, we use the entity linker (Section 3) to identify the entities and numbers in the original sentence Y and replace them with placeholder "[ENT]", which we call as the template Y_T. During the generation of GPT-TabGen, instead of directly predicting the final sentence Y, we first predict the template Y_T and then Y. The process is simply realized by maximizing the overall likelihood of $p(\tilde{Y}|\mathbf{T};\beta)$, where $\tilde{Y} = [Y_T; [\text{SEP}]; Y]$.

Unlike template-based or delexicalized generation (Reiter and Dale, 1997; Wen et al., 2015), which uses rigid slot filling prone to grammatic errors, our fine-grained generation has the flexibility to modify the surface form of non-slot words, which alleviates the linguistic coherence problem (Sharma et al., 2017).

By decoupling sentence structure generation and entity grounding, our proposed coarse-to-fine scheme could partially alleviate the mismatch problem. For example, the generation of "Canada" is now aware of "more than" in the latter part of the sentence, which exposes the model to more context than standard monotonic models to help make logically consistent decisions though the dependency on the "1" and "Mexico" is still not captured. The proposed two-step generation could be viewed as the first step towards a fully non-monotonic generation model to solve such mismatch problem.

6 Experiments

In this section, we explain the experimental details and then comprehensively report the automatic evaluation of different generation models and training algorithms. Finally, we will conduct detailed human evaluation and error analysis.

6.1 Experiment Setup

For the non-pretrained models, we fix the hidden size of both LSTM and transformer to be 256, the transformer is 3-layered with 4 heads, while LSTM is also 3-layered. We use Adam optimizer (Kingma and Ba, 2015) with a learning rate of 2e-4 to jointly optimize the parameters and keep the model with the best perplexity on the validation set. During test time, we use a greedy search to generate text and calculate the BLEU-1,2,3 scores with the 5 references from the table. For the pre-trained models, we base our implementation on Huggingface's Transformer (Wolf et al., 2019) for both BERT (Devlin et al., 2019) and GPT-2 (Radford et al., 2019) with subword unit vocabulary of 30K. During linearization, we found that using the whole table compromises the performance greatly, partly due to 1) over-length issue with pre-trained LM, 2) too much irrelevant information input. Therefore, we propose to use partial table as input, specifically, we run entity linking over the sentences to detect the linked columns of the table and only linearize the partial table as input P_T.

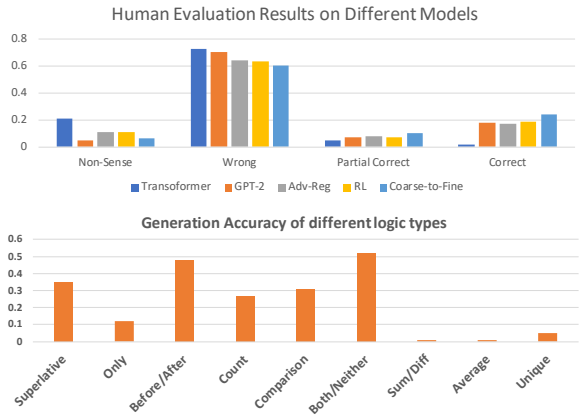

Figure 8: The human evaluation results of different models on the sampled sentences.

Both are finetuned using Adam optimizer (Kingma and Ba, 2015) with a learning rate of 1e-6. In both adversarial training and reinforcement learning algorithms, we add maximum likelihood objective to stabilize the training, we select the appropriate balancing factor based on the validation Adv-Acc socre. For coarse-to-fine training, we first warm up the model to generate the template sequence and then finetune it on the concatenated full sequence. Model selection is based on the bleu-3 score on validation split.

Model	Training	PPL	BLEU-1	BLEU-2	BLEU-3	SP-Acc	NLI-Acc	Adv-Acc
Field-Gating + LSTM	MLE	27.7	42.3	19.5	6.9	38.0	56.8	56.2
Field-Gating + Trans	MLE	26.8	44.1	20.9	8.3	38.5	57.3	58.1
Field-Infusing + LSTM	MLE	27.9	43.1	19.7	7.1	38.6	57.1	56.9
Field-Infusing + Trans	MLE	26.9	43.7	20.9	8.4	38.9	57.3	58.2
BERT-TabGen (sm)	MLE	7.5	47.8	26.3	11.9	42.2	68.1	62.4
GPT-TabGen (sm)	MLE	8.8	**48.8**	**27.1**	12.6	42.1	68.7	62.3
GPT-TabGen (sm)	Adv-Reg	12.1	45.8	23.1	9.6	40.9	68.5	64.7
GPT-TabGen (sm)	RL	11.3	45.1	23.6	9.1	**43.1**	67.7	61.9
GPT-Coarse-to-Fine (sm)	MLE	-	46.6	26.8	**13.3**	42.7	**72.2**	**64.9**
BERT-TabGen (lg)	MLE	6.3	49.1	27.7	13.5	44.4	73.9	64.0
GPT-TabGen (med)	MLE	6.8	**49.6**	**28.2**	14.2	44.7	74.6	64.3
GPT-TabGen (med)	Adv-Reg	10.1	47.2	24.0	10.8	44.1	73.0	65.4
GPT-TabGen (med)	RL	10.0	46.4	24.1	10.0	**45.5**	73.3	63.7
GPT-Coarse-to-Fine (med)	MLE	-	49.0	28.3	**14.6**	45.3	**76.4**	**66.0**

Table 2: The experimental results of different models on the test split of LOGICNLG, where we split the table into non-pretrained LSTM/Transformer, small pre-trained LM (sm) and medium/large pre-trained LM (med/lg).

6.2 Experimental Results

We first perform an automatic evaluation to approximately measure the performance of different models and then conduct an in-depth human evaluation to have a better understanding.

Automatic Evaluation: The experimental results are summarized in Table 2, where we comprehensively survey different architectures and training algorithms. For the non-pretrained models, we observe that Transformer is slightly better than LSTM and two different table encoding strategies achieve similar results. In contrast, pre-trained models are much better at lowering the perplexity, besides the generated sentences significantly outperform the non-pretrained models in terms of both fluency and fidelity score with GPT-TabGen and BERT-TabGen achieving similar performance. As the BERT-TabGen runs much slower due to multiple passes of decoding, we favor GPT-TabGen in the following experiments. With the adversarial regularization and reinforcement training, the model can only improve the optimized fidelity metric, with the fluency scores dropping significantly. Such phenomena confirm our assumption about the caveats of the monotonic generation paradigm. For the proposed coarse-to-fine generation scheme, as the "[ENT]" tokens are replaced by entity names, which normally contain a phrase like "Feb 2nd". Such n-gram phrase substitution preserves the completeness of entity names and thus leads to higher 2/3/4-gram matches, which translates to higher BLEU-3 and lower BLEU-1 in Table 2. The proposed coarse-to-fine generation can yield reasonable improvement over NLI-Acc and Adv-Acc, which demonstrates its advantages of in capturing logical dependency.

Human Evaluation To further investigate the quality of the generated text, we propose to perform human evaluation. Specifically, we sample 200 sentences from different models and distribute them independently to human experts (graduate students from the computer science department) to verify their quality. Specifically, the quality measure is categorized into categories: 1) non-sense: the sentence does not make much sense, which is mainly due to disfluency or repetition problem. 2) wrong: a fluent sentence with wrong logic. 3) partial-correct: the sentence contains more than one fact, at least one of them is correct 4) correct: the high-quality in both fluency and logic correctness. We demonstrate the results in Figure 8, from which we observe that pre-training significantly decreases the non-sense proportion. However, the RL and Adv-Reg both harm the fluency and lead to more non-sense sentences. In contrast, the coarse-to-fine model can maintain the non-sense proportion while significantly increasing correct/partial-correct sentences. From human evaluation, even the best performing model can get slightly over 20% of its prediction logically correct, which reflects the challenges of LOGICNLG for existing paradigm.

Evaluation Metrics We here analyze the effectiveness of the defined automatic evaluation metrics for fidelity evaluation. For the Parsing-based evaluation and NLI-based evaluation, we use the adversarial set (containing positive/negative sample pairs) to evaluate their consistency with human judges. Parsing-based model only achieves an ac-

curacy of 60%, while NLI-based model achieves a higher accuracy of 65%. It indicates that the fidelity measurement model is itself a very challenging problem and the existing models are still in a premature stage. Therefore, the exact number of SP-Acc or NLI-Acc cannot reliably reflect the exact proportion of sentences logically entailed by the table. However, we still believe they are informative for model development based on the following reasons: 1) the automatic fidelity scores are quite stable, not sensitive to random initialization or different configurations, 2) when comparing different models (Transformer vs. GPT-2 vs. RL/Adv-Reg vs. Coarse-to-Fine), the trends of different automatic scores are consistent with human evaluation, which indicates its potential in assisting the development of new models.

Fine-grained Analysis To better understand the generation model's reasoning capability in regarding different logical operations, we pick the most frequent 9 operations (definition in the Appendix) and analyze the best model's capability in expressing these different logic. We demonstrate our human evaluation in Figure 8 to make the following inspections: 1) the model performs best in justifying the order of different entities (before/after) and relating two entities (both/neither/comparison). 2) the model performs reasonably well at superlative and count operation. 3) the generation model performs much worse in operations like "only, unique". 4) the model is not able to perform mathematical aggregation like average, sum, etc. Overall, the string-based operations are easier than numeric-based operations, how to infuse the numeric knowledge is an open research question to move forward.

7 Related Work

Natural Language Generation Natural language generation is a long-standing problem (Kukich, 1983; Holmes-Higgin, 1994; Reiter and Dale, 1997), which involves generating text from records or data. Recently, many neural-based generation models have been proposed (Puduppully et al., 2019a,b; Lebret et al., 2016; Wiseman et al., 2018) to achieve impressive performance on the existing datasets (Chen and Mooney, 2008; Liang et al., 2009; Lebret et al., 2016; Dušek et al., 2019; Wiseman et al., 2017) since the annotated text are mostly surface-level annotation without logical inference. Unlike them, LOGICNLG has rich inference, which poses great challenges to existing models and evaluations.

Non-monotonic Generation There have been attempts recently to study the problem of non-monotonic text generation, which aims to teach the generation model to learn the generation order without external supervision (Ford et al., 2018; Welleck et al., 2019; Gu et al., 2019; Mansimov et al., 2019). These models have shown to learn rational generation order to approach similar performance as the left-to-right case. These approaches are useful at capturing more sophisticated dependency within the sentence, which provides a plausible direction to pursue in LOGICNLG.

Factualness Evaluation Fidelity is an important research topic in generation, In ROTOWIRE (Wiseman et al., 2017) and MSCOCO (Lin et al., 2014), IE-based extractive evaluation (Rohrbach et al., 2018; Dhingra et al., 2019) are adopted for surface-level matching to replace costly human evaluation. In abstractive summarization, Goodrich et al. (2019) proposes NER + Relation Classification method to investigate fidelity in generated summarization while Kryściński et al. (2019) proposes to use NLI models to understand the entailment between generated text with the given document. These evaluations are beyond surface-level to study more sophisticated linguistic phenomena like paraphrasing, compression, entailment, inclusion, etc, which are common in summarization tasks.

8 Conclusion

In this paper, we propose logical NLG to study the logical inference problem in generation. We conduct comprehensive experiments to show the existing NLG models are restricted by its monotonic nature and conclude this to be a proper next-step problem to study NLG systems. There are still some unsolved problems for Logical NLG, e.g. how to improve the quality of automatic metrics to better help human automatically judge models' performances. To promote the research in this direction, we host a LogicNLG challenge[2] to help better benchmark the current progress.

9 Acknowledgement

The authors would like to thank the anonymous reviewers for their thoughtful comments.

[2] https://competitions.codalab.org/competitions/24527

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Steven Bird. 2006. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

David L Chen and Raymond J Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *Proceedings of the 25th international conference on Machine learning*, pages 128–135. ACM.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2019. Tabfact: A large-scale dataset for table-based fact verification. *arXiv preprint arXiv:1909.02164*.

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

Bhuwan Dhingra, Manaal Faruqui, Ankur Parikh, Ming-Wei Chang, Dipanjan Das, and William Cohen. 2019. Handling divergent reference texts when evaluating table-to-text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4884–4895.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *ACL*.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2019. Evaluating the state-of-the-art of end-to-end natural language generation: The E2E NLG Challenge. *arXiv preprint arXiv:1901.11528*.

Nicolas Ford, Daniel Duckworth, Mohammad Norouzi, and George Dahl. 2018. The importance of generation order in language modeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2942–2946.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Ben Goodrich, Vinay Rao, Peter J Liu, and Mohammad Saleh. 2019. Assessing the factual accuracy of generated text. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 166–175. ACM.

Jiatao Gu, Qi Liu, and Kyunghyun Cho. 2019. Insertion-based decoding with automatically inferred generation order. *TACL*.

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

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Paul Holmes-Higgin. 1994. Text generationusing discourse strategies and focus constraints to generate natural language text by kathleen r. mckeown, cambridge university press, 1992, pp 246,£ 13.95, isbn 0-521-43802-0. *The Knowledge Engineering Review*, 9(4):421–422.

Anjuli Kannan and Oriol Vinyals. 2017. Adversarial evaluation of dialogue models. *arXiv preprint arXiv:1701.08198*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *ICLR*.

Wojciech Kryściński, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Evaluating the factual consistency of abstractive text summarization. *arXiv preprint arXiv:1910.12840*.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *Proceedings of the 21st annual meeting on Association for Computational Linguistics*, pages 145–150. Association for Computational Linguistics.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213.

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 91–99. Association for Computational Linguistics.

Percy Liang, Michael I Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Angli Liu, Jingfei Du, and Veselin Stoyanov. 2019. Knowledge-augmented language model and its application to unsupervised named-entity recognition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1142–1150.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Table-to-text generation by structure-aware seq2seq learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Elman Mansimov, Alex Wang, and Kyunghyun Cho. 2019. A generalized framework of sequence generation with application to undirected sequence models. *arXiv preprint arXiv:1905.12790*.

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 on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019a. Data-to-text generation with content selection and planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6908–6915.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019b. Data-to-text generation with entity modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2023–2035, Florence, Italy. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Sainandan Ramakrishnan, Aishwarya Agrawal, and Stefan Lee. 2018. Overcoming language priors in visual question answering with adversarial regularization. In *Advances in Neural Information Processing Systems*, pages 1541–1551.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.

Anna Rohrbach, Lisa Anne Hendricks, Kaylee Burns, Trevor Darrell, and Kate Saenko. 2018. Object hallucination in image captioning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4035–4045.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083.

Shikhar Sharma, Jing He, Kaheer Suleman, Hannes Schulz, and Philip Bachman. 2017. Natural language generation in dialogue using lexicalized and delexicalized data. *ICLR Workshop*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Sean Welleck, Kianté Brantley, Hal Daumé Iii, and Kyunghyun Cho. 2019. Non-monotonic sequential text generation. In *International Conference on Machine Learning*, pages 6716–6726.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721.

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.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2018. Learning neural templates for text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3174–3187.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Ningyu Zhang, Shumin Deng, Zhanlin Sun, Jiaoayan Chen, Wei Zhang, and Huajun Chen. 2019. Relation adversarial network for low resource knowledgegraph completion. *arXiv preprint arXiv:1911.03091*.

A Dataset Examples

In order to give readers a better sense of the statements in LOGICNLG, we demonstrate some typical examples below as Figure 9 and Figure 10. Each table in the dataset is associated with five different examples covering diversified inference skills. For example, Figure 9 requires 'all' operation to identify multiple rows having the same value on certain properties. Figure 10 requires the model to perform superlative, or count operation to identify the numerically highest number.

B Logical Operation Distribution

The dataset consists of the most common types of logical inference in our daily communication, to help the readers understand the semantic meaning of these inference, we list their definition and some examples below:

- superlative: operations involving max,min or other comparison operation to get the lowest or highest value. Sentence: xxx is the tallest player in xxx team.
- only: operation to identify the single entity which has a unique property the other entries do not have. Sentence: xxx is the only person to win all the games.
- before/after: operations to compare time or spatial order. Sentence: xxx is born before xxx.
- count: operations to enumerate the amount of entries meeting certain criterion. Sentence: there are two people from the central united states.
- comparison: operations to compare two or given number of entities. Sentence: xxx has better income than xxx.
- both/neither: operations to summarize the common properties of two entries. Sentence: xxx and xxx are both from the same country.
- sum/diff: operations to perform numeric summation or difference between numbers. Sentence: xxx gives 1 more dollars than xxxx in the donation.
- average: the average number of people attending the game is 500.
- unique: the uniq operation in sql to assemble summarize different entities. Sentence: from the table, players are from 4 unique countries.

C Semantic Parser

Specifically, the scorer is realized by a matching model f_γ, which takes a logic form P and the statement Y to output a consistency score $f_\gamma(P, Y)$ between range of [0,1] with higher value indicating better consistency. As no groundtruth logical forms are provided, we utilize weakly supervised training. The set of logical forms generated is denoted as $\mathbf{P}$, the logical forms returning binary value of `True` is viewed as pseudo positive example $\mathbf{P}^+$ and the logical forms returning `False` is treated as pseudo negative example $\mathbf{P}^-$. We propose to optimize the following objective to discriminate two sets:

$$\arg\max_{\gamma} \mathop{\mathbb{E}}_{(\mathbf{T},Y)\in D_{train}} [\mathop{\mathbb{E}}_{P\in\mathbf{P}^+}[f_\gamma(P,Y)] - \mathop{\mathbb{E}}_{P\in\mathbf{P}^-}[f_\gamma(P,Y)]]$$

As demonstrated in Figure 11, our semantic parser is comprised of three different parts, namely a resolution model, a breadth-first search model and a ranker model. The resolution model will try to figure out what are the entities appearing in the table and what are the numbers it needs to infer. These results are pushed to a buffer as the initial point, then the BFS search will try to compose plausible logical forms based on the values from the buffer. However, most of the synthesized logical forms are not relevant to the semantics the sentence is aimed to express. In the end, we need to train a ranker, which can learn to identify the most consistent logical form and use that to represent the formal semantics of given sentence.

D Qualitative Example

Next, we demonstrate some generated samples in Figure 12, which are generated from a table crawled from Wikipedia page[3]. Though most of the text generated by the model is coherent and reasonable, we do observe some disfluency like repetition, contradiction, erroneous sentences like the sentence 5. For the other sentences, three of them are logically correct, the first sentence contains quite complex logic with three different symbolic operations "argmax, argmin, after". The fourth and sixth sentences involve operations like "filter, count". In contrast, the second and third examples

[3] https://en.wikipedia.org/wiki/2007%E2%80%9308_Golden_State_Warriors_season

player	country	year (s) won	total	to par
larry nelson	united states	1981 , 1987	152	+ 8
jack nicklaus	united states	1963 , 1971 , 1973 1975 , 1980	152	+ 8
lee trevino	united states	1974 , 1984	152	+ 8
hubert green	united states	1985	153	+ 9
lanny wadkins	united states	1977	155	+ 11
dave stockton	united states	1970 , 1976	157	+ 13

larry nelson , jack nicklaus , and lee trevino all shot 8 strokes over par
larry nelson , lee trevino , and dave stockton each won two pga championships in the 1970s - 1980s
jack nicklaus had more pga championship wins than larry nelson and lee tevino combined
dave stockton shot five strokes worse than larry nelson , jack nicklaus , and lee trevino
three golfers shot worse than 8 strokes over par

Figure 9: Example from LOGICNLG.

round	date	opponent	venue	result	attendance
r3	8 january 2002	wycombe wanderers	a	2 - 2	9921
r3r	15 january 2002	wycombe wanderers	h	1 - 0	11894
r4	26 january 2002	york city	a	2 - 0	7563
r5	16 february 2002	walsall	a	2 - 1	8766
qf	10 march 2002	west bromwich albion	a	1 - 0	24811
sf	14 april 2002	chelsea	n	0 - 1	36147

the lowest attendance when fullham won was 7563
fullham fc only played one time at venue h
fullham fc played three times in the month of january
fullham fc faced the wycombe wanderers two times in the month of january
the only defeat of fullham for the 4 first months of 2002 fc was when they face chelsea

Figure 10: Example from LOGICNLG.

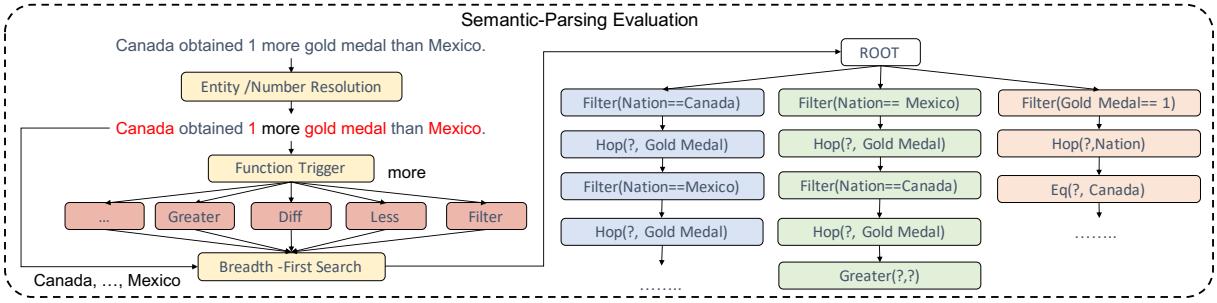

Figure 11: The BFS-based parser used in our evaluation.

are factually incorrect as the team only competes with "Seattle" once and the 3 games are not in a row. We can see that the errors are quite diversified, it is difficult to debug what is the source of these errors by simply looking into the deep generation model. In the future, more interpretable generation model need to be built to make the inference process more transparent.

Title: Golden State Warrior: NBA Season 2007-2008

Date	Visitor	Score	Home	Attendance	Leading Player	Record
12 / 2	golden state warriors	109 - 96	seattle supersonics	11461	stephen jackson	9 - 7
12 / 3	orlando magic	117 - 123	golden state warriors	18527	stephen jackson	9 - 8
12 / 7	miami heat	120 - 113	golden state warriors	19596	stephen jackson	11 - 8
12 / 28	denver nuggets	120 - 124	golden state warriors	20001	stephen jackson	17 - 13
12 / 16	golden state warriors	87 - 109	detroit pistons	22076	matt barnes	13 - 11
12 / 17	golden state warriors	125 - 117	memphis grizzlies	10549	stephen jackson	14 - 11

✓ 1. The game with the lowest in Attendance took place after the game with the highest in Attendance.
✗ 2. The Golden State Warrior played against the Seattle Supersonics 2 time.
✗ 3. The Warrior won 3 game in a row during the 2007 - 08 Season.
✓ 4. The Golden State Warrior lost 2 game when playing at Home.
✗ 5. There were 4 time that was a Leading Scorer, and 4 time that was a Leading Scorer.
✓ 6. Stephen Jackson was the leading scorer 5 different times during the 2007 - 08 Season.

Figure 12: The statements generated by GPT-TabGen model with random sampling.