electronics

MDPI

Article

DiffuD2T: Empowering Data-to-Text Generation with Diffusion

Heng Gong [1], Xiaocheng Feng [1,2] and Bing Qin [1,2,*]

[1] School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China
[2] Peng Cheng Laboratory, Shenzhen 518000, China
[*] Correspondence: qinb@ir.hit.edu.cn

Abstract: Surrounded by structured data, such as medical data, financial data, knowledge bases, etc., data-to-text generation has become an important natural language processing task that can help people better understand the meaning of those data by providing them with user-friendly text. Existing methods for data-to-text generation show promising results in tackling two major challenges: content planning and surface realization, which transform structured data into fluent text. However, they lack an iterative refinement process for generating text, which can enable the model to perfect the text step-by-step while accepting control over the process. In this paper, we explore enhancing data-to-text generation with an iterative refinement process via diffusion. We have four main contributions: (1) we use the diffusion model to improve the prefix tuning for data-to-text generation; (2) we propose a look-ahead guiding loss to supervise the iterative refinement process for better text generation; (3) we extract content plans from reference text and propose a planning-then-writing pipeline to give the model content planning ability; and (4) we conducted experiments on three data-to-text generation datasets and both automatic evaluation criteria (BLEU, NIST, METEOR, ROUGE$_L$, CIDEr, TER, MoverScore, BLEURT, and BERTScore) and human evaluation criteria (Quality and Naturalness) show the effectiveness of our model. Our model can improve the competitive prefix tuning method by 2.19% in terms of a widely-used automatic evaluation criterion BLEU (BiLingual Evaluation Understudy) on WebNLG dataset with GPT-2 Large as the pretrained language model backbone. Human evaluation criteria also show that our model can improve the quality and naturalness of the generated text across all three datasets.

Keywords: diffusion; data-to-text generation; natural language processing; artificial intelligence

check for updates

Citation: Gong, H.; Feng, X.; Qin, B. DiffuD2T: Empowering Data-to-Text Generation with Diffusion. *Electronics* **2023**, *12*, 2136. https://doi.org/10.3390/electronics12092136

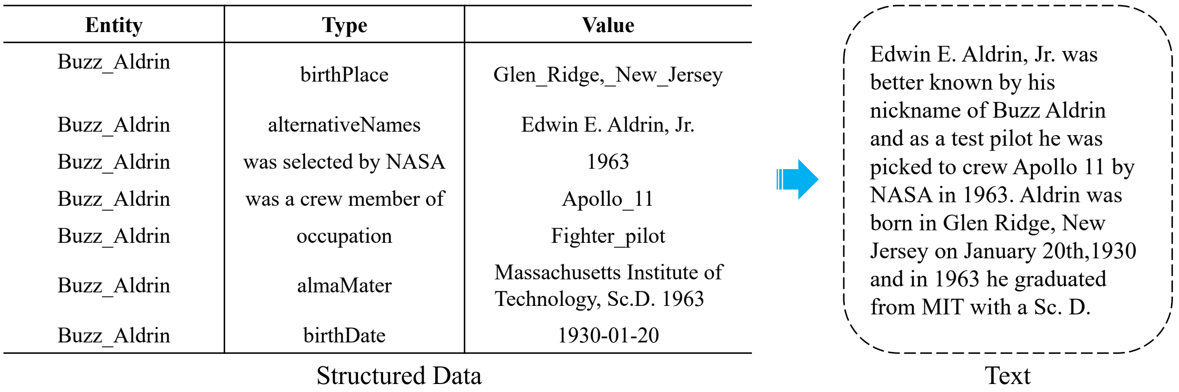
Academic Editor: Rui Pedro Lopes

Received: 3 April 2023
Revised: 1 May 2023
Accepted: 3 May 2023
Published: 7 May 2023

1. Introduction

There are many structured data in real life, such as medical data, financial data, knowledge bases, etc. They pose challenges for users to efficiently understand the important information behind those data, especially when it requires professional knowledge. For example, Gatt et al. [1] found that a textual report of medical data makes it easier for patients to comprehend. Data-to-text generation has become an important task in natural language processing. It processes structured data and automatically generates user-friendly text to help users better understand the data. Figure 1 shows an example of data-to-text generation.

In recent years, the success of data-driven deep learning models [2–4] with large-scale datasets [5–7] has made the end-to-end model become mainstream in the field of data-to-text generation. It faces two challenges in generating high-quality text: content planning [8–11] and surface realization [12,13], which transforms important data into natural language. The former analyzes the structured data, selects important information from the data, and orders them naturally, which makes the unordered data more in line with human reading habits. The latter trains a text generation model to transform the processed structured data into text. After the remarkable performance of a pretrained language model (PLM) [14], using the linguistic knowledge in the PLM and adapting it to the data-to-text generation task became popular. There are two methods: finetuning the PLM on the data-to-text generation dataset [15–17] or prefix tuning [18], which freezes the PLM and prepend some trainable task-specific "virtual tokens" to adapt the PLM to different tasks. Those "virtual tokens" help the PLM understand the requirement of different tasks. For example, the "virtual tokens" for data-to-text generation must guide the PLM to plan the structured data and generate fluent text. The latter shows the potential to reduce the cost of adapting to the downstream tasks while outperforming the method of finetuning PLM for data-to-text generation. However, the methods above generate the text at once, without an iterative refinement process that enables the model to perfect the text step-by-step while accepting any control or supervision during that process. In this paper, we explore using diffusion [19] to improve the modeling of "virtual tokens" for prefix tuning on data-to-text generationso that it can generate text of better quality.



Entity	Type	Value
Buzz_Aldrin	birthPlace	Glen_Ridge,_New_Jersey
Buzz_Aldrin	alternativeNames	Edwin E. Aldrin, Jr.
Buzz_Aldrin	was selected by NASA	1963
Buzz_Aldrin	was a crew member of	Apollo_11
Buzz_Aldrin	occupation	Fighter_pilot
Buzz_Aldrin	almaMater	Massachusetts Institute of Technology, Sc.D. 1963
Buzz_Aldrin	birthDate	1930-01-20

Edwin E. Aldrin, Jr. was better known by his nickname of Buzz Aldrin and as a test pilot he was picked to crew Apollo 11 by NASA in 1963. Aldrin was born in Glen Ridge, New Jersey on January 20th,1930 and in 1963 he graduated from MIT with a Sc. D.

Figure 1. An illustration of an example for data-to-text generation. In this example, the structured data are seven related triples from the knowledge base. Each triple consists of the name of the entity, the type of this information, and the corresponding value. Given the structured data, a text report faithfully expresses the information in the data.

Recently, diffusion models [19,20] have shown great potential in generating high-quality examples in the domain of image [21,22] and audio [23,24] generation. They follow a new paradigm of the noising and denoising process. The former corrupts data with Gaussian noise, while the latter reconstructs the data from pure Gaussian noise step-by-step. With the refinement process, the diffusion models are able to improve the output gradually, and generate high-fidelity text.

However, there is a fundamental difference between image generation and text generation; the former deals with continuous data, while the latter processes discrete textual data. Some works [25–27] try to bridge the gap between continuous and discrete data by exploiting diffusion models on the continuous embedding space, then mapping them to the discrete vocabulary to generate text. Although they show encouraging results in generating text non-autoregressively, the rounding procedure of converting diffusion-empowered embedding space to a discrete vocabulary limits the representation power of the diffusion models. The diffusion-based generation model does not perform well on data-to-text generation, based on our preliminary study.

In this paper, we empower the representation of the continuous prefixes (the "virtual tokens") with the step-by-step refinement process of the diffusion model to help PLM better understand the requirement of tasks. In addition, we address two problems in this process: how to provide a meaningful supervision signal for the refinement process and how to give the model content planning ability. For the first question, we propose a look-ahead guiding loss. For the denoised prefix of each diffusion step, we employ the denoising step one more time to obtain a look-ahead denoised prefix. We require that the look-ahead denoised prefix should perform better (i.e., lower loss for PLM) with the help of hinge loss. As for the second question, we extract key phrases about data from the reference text and split the training process into two stages: content planning and surface realization. The prefixes for learning content planning are used to initialize the prefixes for text generation, giving the model content planning ability before training them to generate text.

We validated our proposed models on three data-to-text generation datasets, including WebNLG, E2E, and DART. The results and case study show the proposed modules' superiority over prefix tuning and other competitive methods across different datasets. We have four main contributions:

- We propose to improve prefix tuning for data-to-text generation with the step-by-step refinement diffusion model.
- We propose a look-ahead guiding loss to supervise the refinement process of the diffusion model.
- We propose a planning-then-writing training pipeline to provide the model content planning ability.
- We conducted experiments and analyses on three data-to-text generation datasets and both automatic evaluation criteria (Sections 5.2 and 5.5) and human evaluation criteria (Section 5.7) show the effectiveness of our model.

2. Literature Review

2.1. Data-to-Text Generation

In recent years, with the help of high-quality datasets [5–7] and sequence-to-sequence architecture [2–4], data-driven neural data-to-text generation models [8,9,28] have become mainstream and achieved remarkable performance. Traditionally [29], data-to-text generation models need to address two major problems: content planning and surface realization, which is used to generate the final text. Many researchers [8–11] try to model content planning in neural data-to-text generation models explicitly. The first work proposed a two-stage generation pipeline that uses an attention-mechanism-based pointer network [30] to select and plan important information from structured data, and then generate the final text using another encoder–decoder model. The second one creates memory slots to track and update the entity's representation for the model to plan entities dynamically during generation. The third one considers the previously generated text when planning which data should be mentioned. The final one explores planning at a higher level by creating paragraph plans for the entities. Others focus on improving the quality of the generated text [12,13]. The first one defines some executable mathematical operations on structural data to include more source information for generating more accurate text. The second one utilizes a graph neural network [31] to strengthen the encoding of structural data.

In light of the success of the pretrained language model [14], adapting the pretrained language model to data-to-text generation has gradually become popular. There are two types of methods for that: finetuning the whole pretrained language model [15] or prefix learning [18]. As for the first direction, some explore enhancing the reasoning ability for data-to-text generation [32–34]. The first work proposes a new data-to-text generation dataset that requires the model to perform logical inference when generating text. The second work manually annotates the inference process with logical forms, enabling the model to learn how to perform inference. The last work explores the data-to-text generation with logical forms in a few-shot setting and uses self-training to enlarge the training corpus with generated logical forms gradually. Some explore data-to-text generation in few-shot/zero-shot setting [16,17,35]. The first work exploits the knowledge in pretrained language model to finetune a data-to-text generation model with limited training instances in the target domain. The second work takes a different path by using a language model to generate new text to augment the limited corpus. The last work takes inspiration from the traditional pipeline-style data-to-text system, including stages of ordering, aggregation, and paragraph compression, and uses pretrained language models to implement these stages without parallel data-text training pairs. Regarding prefix tuning [18], it provides an efficient training paradigm for utilizing the pretrained language model. Specifically, it freezes the parameters of the pretrained language model and only tunes a set of prepended trainable continuous vectors to the language model, which consists of much fewer parameters. These vectors can be considered the "soft" descriptions of the task requirements to adapt the language model to the given task. Based on this, Clive et al. [36] proposes to include some input-dependent information to the prefix in addition to the original task-specific prefix, and Chen et al. [37] proposes to bridge the prompt learning and adapter learning. While pretrained language models achieve good performance for data-to-text generation, they generate the final text in a single run without the iterative refinement of the generated text. The iterative refinement process enables the model to accept control during generation and correct potential mistakes. Since diffusion models' [19,20] iterative refinement process shows great potential in the image and audio generation, we explore using the diffusion model to boost the performance of data-to-text generation.

Table 1 shows the comparison of significant studies in data-to-text generation in terms of their contributions, datasets, and models' performance based on BLEU score.

Table 1. Comparisons of data-to-text generation models. We report the evaluation criterion BLEU (BiLingual Evaluation Understudy) score on the dataset in **bold**.

Author	Contributions	Dataset	BLEU
Chen et al. [15]	They propose to pretrain the data-to-text model with large-scale unlabeled text and knowledge graph.	**E2E** [5], WebNLG [6], WikiBio [38]	68.05
Chang et al. [17]	They propose to use the pretrained language model to generate new text to augment the limited corpus.	**E2E** [5], WebNLG [6]	68.88
Li and Liang [18]	They propose to finetune lightweight task-specific parameters while freezing the parameters of pre-trained language model during training.	**E2E** [5], WebNLG [6], DART [7]	70.30
Hu et al. [39]	They propose to insert trainable rank decomposition matrices into each layer of the model and freeze the parameters of pretrained language model (PLM) to reduce the cost of finetuning.	**E2E** [5], WebNLG [6], DART [7]	70.40
Chen et al. [33]	They propose to manually annotate the logical inference process with logical forms and train the model to do logical inference explicitly.	**Logic2Text** [33]	31.44
Nie et al. [12]	They propose to execute mathematical operations on structural data to provide more data for generating more accurate text.	**RotoWire** [28], ESPN [12]	14.74
Puduppully and Lapata [11]	They propose to create paragraph plans for entities.	**RotoWire** [28], MLB [9]	15.46
Puduppully et al. [8]	They propose a two-stage generation pipeline that selects and plans important information from structured data, and then generates the final text.	**RotoWire** [28]	16.50
Puduppully et al. [9]	They propose to use memory slots to track the entity's representation during generation for entity planning.	**RotoWire** [28], MLB [9]	16.12
Chen et al. [10]	They propose to consider the previously generated text when planning the next structured data.	**RotoWire** [28], NBAZhn [10]	16.38

2.2. Diffusion Model

Recently, the diffusion model [19,20] has become a new paradigm with great potential in generating high-quality examples in image [21,22] and audio generation [23,24]. During inference, diffusion models employ an iterative refinement process to denoise and construct high-quality images or audio from Gaussian noise step-by-step. Nichol et al. [21], Dhariwal and Nichol [22] show the generation diversity of the diffusion models. In light of diffusion's success in generating high-quality images and audio, some researchers explore how to generate high-quality text via diffusion. However, there is a fundamental difference between images and text. The images are represented in continuous space, which can be naturally modeled by the diffusion model. However, the generation of words in the text, which uses the softmax function to pick a word with high probability from the vocabulary, is discrete. The adaption of the diffusion model is not intuitive. To bridge the above gap, Li et al. [25], Gong et al. [26], Gao et al. [40] first map each discrete word into the continuous trainable word embedding (vectors), and then use the diffusion model. During inference, after denoising and constructing the high-quality vector representation for each word, they use a trainable linear transformation function, followed by softmax function, to map those vector representations to the probability of generating each word from the vocabulary. Li et al. [25] proposes a diffusion-based language model that can be combined with control signals from a classifier for controllable text generation. Gao et al. [40] analyzes some of the limitations of [25] and proposes three techniques to boost the performance. Gong et al. [26] proposes a sequence-to-sequence diffusion-based text generation model to adapt to conditional text generation tasks, such as summarization, which needs to generate text based on the user's input text, making it the closest work to our model. In this paper, we combine the power of the diffusion model in continuous space directly with prefix tuning for data-to-text generation, while the diffusion-based generation model above does not perform well, based on our preliminary study. In addition, to better adapt to the data-to-text generation task, we introduce look-ahead guiding loss to guide the diffusion process and planning-then-writing pipeline to mimic two core tasks of data-to-text generation: content planning and surface realization.

Table 2 shows the comparison of significant studies of diffusion models in terms of their domain, contributions, datasets, and models' performance based on BLEU score. Since BLEU is designed to evaluate text, Nichol et al. [21] and Kong et al. [23] explore image generation and audio generation, and they do not have the BLEU score.

Table 2. Comparisons of diffusion models. We report the evaluation criterion BLEU (BiLingual Evaluation Understudy) score on the dataset in **bold**. Since BLEU is designed to evaluate text, the first two models on the image or audio generation do not have the corresponding results.

Author	Domain	Contributions	Dataset	BLEU
Nichol et al. [21]	Image	They explore the diffusion model for text-conditional image synthesis and found that the classifier-free guidance performs better.	GLIDE (filtered) [21]	N/A
Kong et al. [23]	Audio	They propose to use the diffusion model for conditional and unconditional audio generation.	LJ [41], Commands [42]	N/A
Li et al. [25]	Text	They propose a diffusion-based language model for controllable text generation.	**α-NLG** [43]	7.1
Gong et al. [26]	Text	They propose a sequence-to-sequence diffusion-based text generation model for conditional text generation.	**Jiang et al. [44]**, Quasar-T [45]	36.22
Gao et al. [40]	Text	They propose three techniques to mitigate the limitation of the diffusion-based language model [25].	**WMT-14** [46], Gigaword [47]	27.23

3. Background

3.1. Task Formulation

Generally, text generation tasks take source document S as input and generate desired text E. For example, for the data-to-text task, the input is a set of structured data, and the desired output is a summary of important information in the data. Given the training corpus, the goal is to train a text generator model $f_\theta(E|S)$. For the data-to-text task, the model $f_\theta(E|S)$ needs to understand the structured information in the data, plan the unordered data naturally and describe them in natural language.

3.2. Diffusion Model

In the domain of images, Ho et al. [20] proposes an efficient Denoising Diffusion Probabilistic Model (DDPM) to produce a high-quality image through a parametrized Markov Chain. It contains two major processes: the forward process, which adds Gaussian noise to the example gradually, and the reverse process, which reconstructs the example from Gaussian noise.

Assuming the data example is y_0, the forward process gradually adds noise to y_0, creating intermediate variables $y_1, y_2, \ldots, y_T$. With sufficient steps T, the final variable y_T follows a standard Gaussian distribution $\mathcal{N}(0, I)$. This step-by-step noising process can be formalized as follows:

$$q(y_{1:T}|y_0) = \prod_{t=1}^{T} q(y_t|y_{t-1}) \tag{1}$$

For each $q(y_t|y_{t-1})$, it samples y_t from a Gaussian distribution as follow:

$$q(y_t|y_{t-1}) = \mathcal{N}(y_t; \sqrt{1-\beta_t} y_{t-1}, \beta_t I) \tag{2}$$

The hyper-parameters in the equation are set manually, and are not trainable parameters in the forward process.

As for the reverse process, given a Gaussian noise sampled from standard Gaussian distribution, it attempts to reconstruct the data step-by-step:

$$p_\theta(y_{0:T}) = p(y_T) \prod_{t=1}^{T} p_\theta(y_{t-1}|y_t) \tag{3}$$

Each step of predicting y_{t-1}, given the noise y_t from the previous step, also follows the Gaussian distribution as defined below:

$$p_\theta(y_{t-1}|y_t) = \mathcal{N}(y_{t-1}; \mu_\theta(y_t, t), \sigma_t^2 \mathbf{I}) \tag{4}$$

Since the data example y_0 is not known during the reverse process or inference, the $\mu_\theta(y_t, t)$ is modeled by a neural network for prediction.

For training the parameters in the reverse process, Ho et al. [20] devise a simplified objective function as follows:

$$L_{simple}(y_0) = \sum_{t=1}^{T} \mathop{\mathbb{E}}_{q(y_t|y_0)} ||\mu_\theta(y_t, t) - \hat{\mu}(y_t, y_0)||^2 \tag{5}$$

This training process draws supervision signal $\hat{\mu}(y_t, y_0) = \frac{\sqrt{\hat{\gamma}_{t-1}}\beta_t}{1-\hat{\gamma}_t} y_0 + \frac{\sqrt{\gamma_t}(1-\hat{\gamma}_{t-1})}{1-\hat{\gamma}_t} y_t$ from the mean of the posterior $q(y_{t-1}|y_0, y_t)$ of the forward process, where $\gamma_t = 1 - \beta_t$ and $\hat{\gamma}_t = \prod_{c=1}^{t} \gamma_c$. Since the forward process contains no trainable parameter, this can be obtained through the Gaussian process.

Based on the analysis by Li et al. [25], using $L_{simple}(y_0)$ to supervise the training of the diffusion model lacks the modeling of the structure of y_0. Instead, they propose to directly parameterize the reverse process by predicting y_0 as follows:

$$L_{reverse}(y_0) = \sum_{t=1}^{T} \mathop{\mathbb{E}}_{y_t} ||f_\theta(y_t, t) - y_0||^2 \tag{6}$$

During inference, for each reverse (denoising) time step t, the diffusion model first predicts y_0 through the trained $f_\theta(y_t, t)$, then sample the y_{t-1} through applying $q(y_{t-1}|y_{t-2})$ iteratively.

4. Methods

As illustrated in Section 1, (1) we use diffusion to improve the data-to-text generation model. (2) We propose look-ahead guiding loss to supervise the refinement process for better text generation. (3) We propose the planning-then-writing pipeline to provide the model content planning ability.

4.1. DiffuD2T: Diffusion for Data-to-Text Generation

As illustrated in Section 1, we use prefix tuning as the base model for data-to-text generation. In this paradigm, prefixes can be considered as "virtual tokens" that are prepended to the pretrained language model. They consist of trainable parameters and serve as task manuals that help the pretrained language model (PLM) adapt to different tasks. The parameters in PLM are frozen during training. To boost the representation power of the prefixes, we use the step-by-step refinement process of the diffusion model. For example, given the structured data <Buzz_Aldrin | birthPlace | Glen_Ridge> (Figure 2), DiffuD2T has three steps: (1) it samples noise from the Gaussian distribution $\mathbb{N}(0, I)$ and obtains a high-quality latent representation y_0 for prefix through the iterative refinement process step-by-step; (2) it transforms the latent representation y_0 into a prefix that matches the shape of the pretrained language model (PLM); and (3) the PLM takes the prefix and the structured data <Buzz_Aldrin | birthPlace | Glen_Ridge> as input and generates the text "Aldrin was born in Glen Ridge." word-by-word.

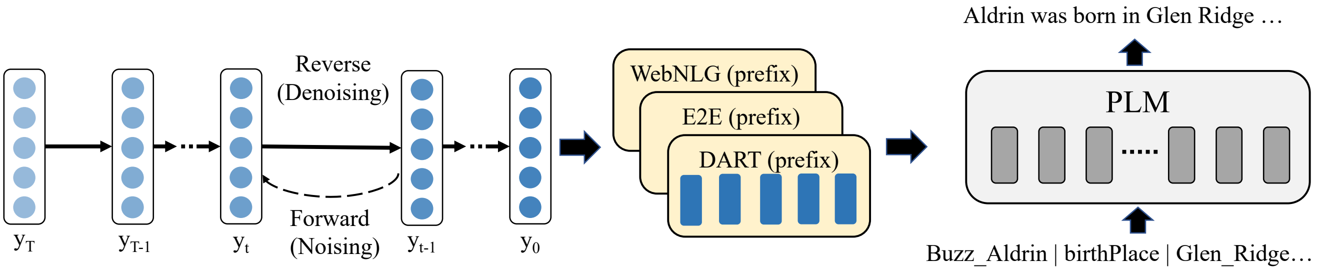


Figure 2. An illustration of the step-by-step optimizing process of the DiffuD2T. The left of the figure presents the iterative refinement process of the diffusion model for the representation of prefixes, as illustrated in Section 3.2. y_T is random noise sampled from the Gaussian distribution $\mathbb{N}(0, I)$. Through the reverse process that denoises y_T step-by-step, we ultimately obtain a high-quality y_0 for the representation of the prefix. The forward process adds noise to the y_0 step-by-step. After getting y_0, we use linear transformation to map it to the shape of PLM so that it can serve as "virtual tokens" that help PLM adapt to different tasks. The parameters for the diffusion model and the linear transformation are trainable while the parameters of PLM are frozen during training. The PLM takes the structured data as input and generates the text with the help of prefixes.

Specifically, after sampling noise y_T from the standard Gaussian distribution $\mathbb{N}(0, I)$, we gradually denoise y_T to y_0 through the chain of denoising function $p_\theta(y_{t-1}|y_t)$, which is defined in Section 3.2. Then, we obtain the $prefix$ as follows. It consists of a linear transformation where W_a is the trainable parameter that will transform the shape of y_0 to the shape of PLM.

$$prefix = W_a y_0 \tag{7}$$

Then, the prefix is used to generate text through the text generator with the probability $p(E|[prefix, S])$.

We propose to modify the training objective function of the diffusion model to accompany both the training of the diffusion model and the training of the text generator as follows:

$$L_{joint}(y_0) = L_{reverse}(y_0) + L_{gen}(y_0) \tag{8}$$

where $L_{reverse}(y_0)$ is defined in Section 3.2 and $L_{gen}(y_0)$ is defined as follows:

$$L_{gen}(y_0) = -\frac{1}{Z}\sum_{g=1}^{G}\sum_{t=1}^{M_g}\log p(E_{g,t}|E_{g,<t},[prefix, S_g]) \tag{9}$$

M_g represents the length of the text, while G represents the number of batches. Z is the normalization factor. The PLM takes both prefix and structured data S_g as input and generates the target text sequence E_g word-by-word. The t in the $E_{g,t}$ denotes the t-th word in the sequence E_g. The training goal is to maximize the text generator's probability on the target text.

4.2. Look-Ahead Guiding Loss

The refinement process of the diffusion model allows control over the denoising process. As illustrated in Figure 3, the prefix can have attributes of our desire by supervising the denoising process. An intuitive way of supervising this process is to guide the denoised representation y_{t-1} to perform better than the y_t in terms of generating high-quality text for the given input.

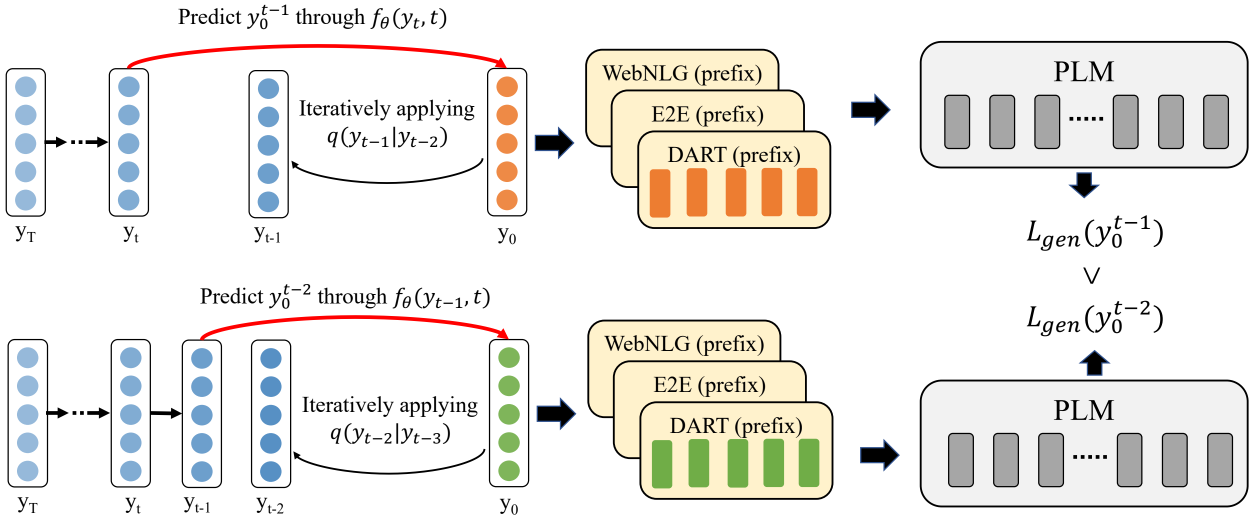


Figure 3. An illustration of our proposed look-ahead guiding loss. Please be reminded that during each denoising step t, we first predict the y_0^{t-1} directly through $f_\theta(y_t, t)$, then obtain the y_{t-1} through the forward process by iteratively applying $q(y_{t-1}|y_{t-2})$, according to Section 3.2. The top of the figure shows that we use the predicted y_0^{t-1} to obtain the corresponding prefix and calculate the loss $L_{gen}(y_0^{t-1})$ of PLM on the target text, which indicates the denoised y_{t-1}'s performance on generating text. Then, as shown in the bottom part of the figure, we take one step further to denoise y_{t-1} again to obtain y_0^{t-2} and y_{t-2}. Similarly, we obtain its performance on generating text $L_{gen}(y_0^{t-2})$. Since the lower the loss, the better the performance, we propose a new look-ahead guiding loss L_{ahead} to supervise $L_{gen}(y_0^{t-2}) < L_{gen}(y_0^{t-1})$. That is, the denoised representation should obtain better performance in generating text than its previous one.

Therefore, we propose the look-ahead guiding loss during training to confine the denoising process. The general idea is that after obtaining the denoised y_{t-1}, we take one step further to denoise the predicted y_{t-1} into y_{t-2}. As described in Section 3.2, during each step of denoising, the model first predicts the y_0 at time step t with y_t directly through $f_\theta(y_t, t)$, and then obtains the y_{t-1} through applying $q(y_{t-1}|y_{t-2})$ iteratively. So, when we denoise y_t to obtain y_{t-1}, we can naturally obtain y_0^{t-1}, which refers to the predicted y_0 at the denoising timestep t. Similarly, when we take one step further to predict y_{t-2} with y_{t-1}, we can also obtain the corresponding y_0^{t-2}. Then, we map the y_0^{t-1} and y_0^{t-2} to the corresponding prefixes with Equation (7). Afterward, we use a text generator to obtain the corresponding loss on the batch data as $L_{gen}(y_0^{t-1})$ and $L_{gen}(y_0^{t-2})$, respectively. L_{gen} is defined in Section 4.1. The idea is that the further denoised representation should perform better on the text generator than the noisier one, that is $L_{gen}(y_0^{t-2}) < L_{gen}(y_0^{t-1})$. For example, given the prefix generated by y_0^{t-2} and the structured data <Buzz_Aldrin | birthPlace | Glen_Ridge>, the text generator should have a higher probability to generate the reference text "Aldrin was born in Glen Ridge" rather than given the prefix generated by y_0^{t-1}. We use the hinge loss to supervise this process as follows. ξ is the expected gap between $L_{gen}(y_0^{t-2})$ and $L_{gen}(y_0^{t-1})$.

$$L_{ahead}(t) = \max(0, \xi - L_{gen}(y_0^{t-1}) + L_{gen}(y_0^{t-2}))) \tag{10}$$

In this way, we guide the diffusion model to progressively denoise the representation to generate better text.

4.3. Planning-Then-Writing Pipeline

To give the model content planning ability, we first extract the content plan from the target text. For words that appear in both the target text and source table, we assume them to be the important information to be described and keep their order in reference text to produce a content plan. Since structured data replace the space between words with "_", in order to match between text and structured data, we remove those "_". Additionally, we allow partial match of the phrases in structured data to increase the coverage of the extracted content plan. As illustrated in Figure 4, during the first stage, we train a content planner with the diffusion-based prefix tuning method DiffuD2T and the look-ahead guiding loss described in Sections 4.1 and 4.2. For example, given the structured data <Buzz_Aldrin | birthPlace | Glen_Ridge> and <Buzz_Aldrin | alternativeNames | Edwin E. Aldrin, Jr.>, the content planner is trained to generate the content plan "Content plan are: Edwin E. Aldrin, Jr.; Buzz Aldrin; Glen Ridge". This content plan puts the nickname of Edwin E. Aldrin, Jr. before his birthplace, which follows the reference text's style of writing ("Edwin E. Aldrin, Jr. was better known by his nickname of Buzz Aldrin ... Aldrin was born in Glen Ridge..."). This model is trained to take the source data as input and plan important information for the task. This also fits the paradigm, which traditionally splits the data-to-text generation tasks into two stages: content planning and surface realization.

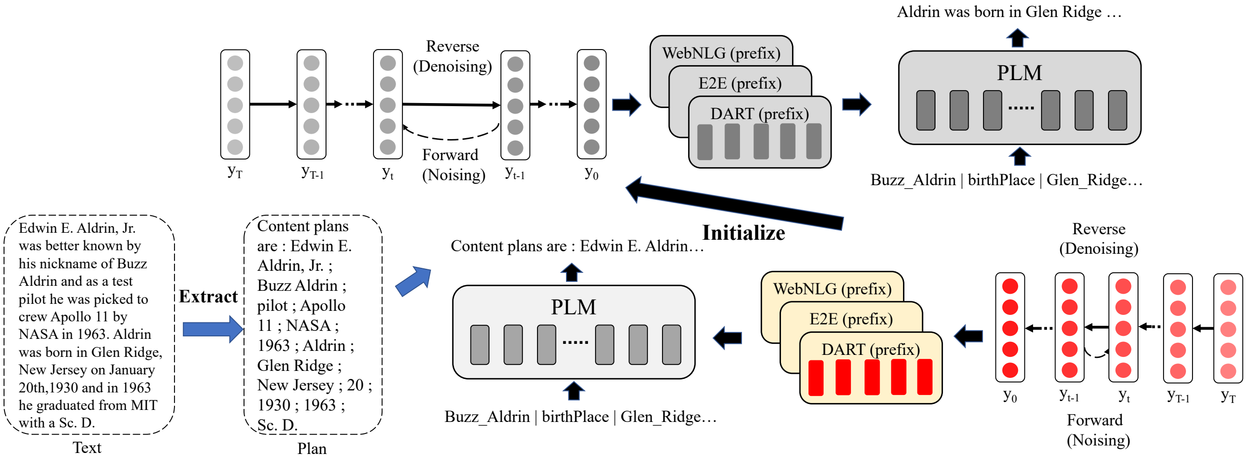

Aldrin was born in Glen Ridge …
Reverse
(Denoising)
y_T
y_{T-1}
y_t
Forward
(Noising)
y_{t-1}
y_0
WebNLG (prefix)
E2E (prefix)
DART (prefix)
PLM
Buzz_Aldrin | birthPlace | Glen_Ridge…
Initialize
Edwin E. Aldrin, Jr. was better known by his nickname of Buzz Aldrin and as a test pilot he was picked to crew Apollo 11 by NASA in 1963. Aldrin was born in Glen Ridge, New Jersey on January 20th,1930 and in 1963 he graduated from MIT with a Sc. D.
Text
Extract
Content plans are : Edwin E. Aldrin, Jr. ; Buzz Aldrin ; pilot ; Apollo 11 ; NASA ; 1963 ; Aldrin ; Glen Ridge ; New Jersey ; 20 ; 1930 ; 1963 ; Sc. D.
Plan
Content plans are : Edwin E. Aldrin…
PLM
Buzz_Aldrin | birthPlace | Glen_Ridge…
WebNLG (prefix)
E2E (prefix)
DART (prefix)
Reverse
(Denoising)
y_0
y_{t-1}
y_t
y_{T-1}
y_T
Forward
(Noising)

Figure 4. An illustration of the planning-then-writing pipeline. The bottom of the figure shows the planning stage. We extract important information from the text to construct the content plan by identifying words that both appear in the target text and the structured data. Then, we use the structured data and the extracted plan to train a content planner with the same model structure as the text generator, described in Sections 4.1 and 4.2. Then, we use the parameters of the content planner to initialize the training of the text generator, which is at the top of this figure in gray, to help the text generator produce better text with the ability to plan.

Then, we use the parameters of the content planner to initialize the training of the final text generation model. Intuitively, the final model is first taught about how to plan the structured data, and then it uses this kind of knowledge to learn further about how to transform the data into text.

4.4. Training and Inference

4.4.1. Training

Given a set of structured data, we linearized them into natural language $S = (s_1, s_2, \cdots, s_m)$, and the model needs to learn to generate the target text $E = (e_1, e_2, \cdots, e_n)$. During training, for each batch of data, the training loss function is $L_{joint}(y_0) + L_{ahead}(t)$ under the framework of multi-task training. There are two losses in the whole training loss $L_{joint}(y_0)$, as defined in Equation (8). For each batch, we sample a timestep t from $1, 2, \cdots, T$ to calculate the loss of $L_{reverse}(y_0^{t-1})$ and use the batch of <structured data, reference text> pair to calculate $L_{gen}(y_0^{t-1})$. The sampled timestep t is also used for calculating the look-ahead guiding loss $L_{ahead}(t)$.

4.4.2. Inference

Since the prefixes are task-specific instead of instance-specific, we only need to perform inference on the diffusion model to obtain the prefixes once to evaluate each task. Specifically, we sample noise y_T from the standard Gaussian distribution $\mathbb{N}(0, I)$, and gradually denoise y_T to y_0 through denoising function $p_\theta(y_{t-1}|y_t)$. In this way, unlike other diffusion-based text generation methods, our method introduces little overhead for evaluating one task. To alleviate the randomness in sampling during the whole process of denoising, we sample 10 times for each task and use the average of the 10 prefixes as the final prefix.

4.5. Algorithm

We illustrate the iterative algorithm for model training in Algorithm 1. First, we pretrain the model with the $L_{joint}(y_0) + L_{ahead}(t)$ on the content planning dataset. For example, given the structured data <Buzz_Aldrin | birthPlace | Glen_Ridge> and <Buzz_Aldrin | alternativeNames | Edwin E. Aldrin, Jr.>, the content planner is trained to arrange the structured data naturally and generate the content plan "Content plan are: Edwin E. Aldrin, Jr.; Buzz Aldrin; Glen Ridge.". We regard the order of structured data in reference text ("Edwin E. Aldrin, Jr. was better known by his nickname of Buzz Aldrin ... Aldrin was born in Glen Ridge ...") as the gold standard for content plans. Then, we use it to initialize the training of the text generator. At epoch o, We apply the diffusion-based prefix tuning method and jointly optimize over the diffusion model loss, look-ahead guiding loss, and text generator loss. For example, given the structured data <Buzz_Aldrin | birthPlace | Glen_Ridge>, we use a diffusion-based prefix tuning method to learn the best prefix that can help the pretrained language model (PLM) to maximize its probability to generate reference text "Aldrin was born in Glen Ridge".

Algorithm 1 Our approach DiffuD2T for data-to-text generation

```
Require: $\mathbb{D}_{\text{train}}$ (The whole training dataset consists of structured data and text),
    $\mathbb{D}_{\text{contentplan}}$(The whole training dataset consists of structured data and extracted plan)
Ensure: $\theta_N$ (The parameters of the whole DiffuD2T model)
Pre-train $\theta_N$ on $\mathbb{D}_{\text{contentplan}}$ for multiple epochs
for epoch $o = 1$ to $N$ do
    for each batch $\mathbb{D}_{\text{batch}}$ in $\mathbb{D}_{\text{train}}$ do
        sample a timestep t from $1, 2, \cdots, T$.
        use diffusion model to denoise $y_t$ into $y_{t-1}$ and calculate the corresponding text
        generation loss $L_{gen}(y_0^{t-1})$
        further denoise the predicted $y_{t-1}$ into $y_{t-2}$ and calculate the corresponding text
        generation loss $L_{gen}(y_0^{t-2})$
        Minimize $L_{joint}(y_0^{t-1}) + L_{ahead}(t)$ with Equations (8) and (10)
    end for
end for
```

5. Experiments

5.1. Dataset

Following Li and Liang [18], we conducted experiments on three standard data-to-text datasets, namely E2E [5], WebNLG [6] and DART [7]. We describe the details of these datasets below.

- The WebNLG dataset focuses on generating textual descriptions for 14 domains. The structured data can be viewed as <entity, type, value> triple. Unlike the other two datasets, it explicitly evaluates the model's generalization performance on unseen domains; that is, the domain is not covered during training. It consists of 18,025, 870, and 1862 instances for training, validation, and test set. The average input length is 49.6 and the average output length is 30.7. Following the official evaluation script (https://github.com/Yale-LILY/dart, accessed on 27 December 2022), we report BLEU [48], METEOR (MET) [49], and TER [50] scores in this paper.
- The DART dataset focuses on open-domain data-to-text generation based on their structured data. The structured data can be viewed as <entity, type, value> triple. It consists of 62659, 2721, and 4159 instances for training, validation, and test set. The average input length is 38.8 and the average output length is 27.3. Following the official evaluation script (https://github.com/Yale-LILY/dart, accessed on 27 December 2022), we report BLEU [48], METEOR (MET) [49], TER [50], MoverScore [51], BERTScore [52] and BLEURT [53] scores in this paper.
- The E2E dataset focuses on generating restaurant descriptions based on their attributes. The structured data can be viewed as <attribute, value> pairs. It consists of 42,061, 547, and 630 instances for training, validation, and test set. The average input length is 28.5, and the average output length is 27.8. Following the official evaluation script (https://github.com/tuetschek/e2e-metrics, accessed on 27 December 2022), we report BLEU [48], NIST [54], METEOR (MET) [49], ROUGE-L (R-L) [55] and CIDEr [56] scores in this paper.

5.2. Automatic Evaluation Criteria

- BLEU [48]: Based on n-gram matching, BLEU assesses the model's performance between the generated text and reference text. BLEU is calculated as follows:

$$P_n = \frac{\sum\limits_{C\in\{\text{GeneratedText}\}} \sum\limits_{\text{n-gram}\in C} \text{Count}_{\text{clip}}(\text{n-gram})}{\sum\limits_{C'\in\{\text{GeneratedText}\}} \sum\limits_{\text{n-gram}'\in C'} \text{Count}(\text{n-gram}')} \tag{11}$$

$$\text{BrevityPenalty} = \begin{cases} 1 & g > r \\ e^{(1-\frac{r}{g})} & g \le r \end{cases} \tag{12}$$

$$\text{BLEU} = \text{BrevityPenalty} \cdot \exp(\sum_{n=1}^{N} w_n \log P_n) \tag{13}$$

Equation (13) has three components. The n-gram precision of the generated text compared to the reference text (Equation (11)). The weight w_n for each n-gram is positive. The brevity penalty (Equation (12)) penalizes generated text that is shorter than the reference text. g is the length of the generated text and r is the length of the reference text. Following Papineni et al. [48], we use BLEU with $N = 4$ as the criteria.
- NIST [54]: Although it is still an n-gram-based metric similar to BLEU, it introduces information-weighted n-gram precision that favors those more informative n-grams. NIST is calculated as follows:

$$Info(w_1 \ldots w_n) = \log_2 \left(\frac{\text{the number of occurrences of } w_1 \ldots w_{n-1}}{\text{the number of occurrences of } w_1 \ldots w_n} \right) \tag{14}$$

$$\text{NIST} = \sum_{n=1}^{N} \frac{\sum\limits_{\substack{\text{co-occurrence of} \\ \text{all } w_1 \ldots w_n}} Info(w_1 \ldots w_n)}{\sum\limits_{\substack{\text{all } w_1 \ldots w_n \\ \text{in generated text}}} (1)} \cdot \exp\left\{ \beta \log^2 \left[\min\left(\frac{L_{gen}}{\overline{L}_{ref}}, 1 \right) \right] \right\} \tag{15}$$

The first fraction of Equation (15) is the information-weighted (Equation (14)) n-gram precision of the generated text compared to the reference text. It gives more weight to those n-grams that occur less frequently, which are considered more informative. L_{gen} is the length of the generated text and $\overline{L}_{ref}$ is the average length of the reference texts. $\frac{L_{gen}}{\overline{L}_{ref}}$ is used to penalize short text generated by the model. $N = 5$, and β is chosen to make this brevity penalty factor as 0.5 when $\frac{L_{gen}}{\overline{L}_{ref}} = \frac{2}{3}$.

- METEOR (MET) [49]: While N-gram matching tends to perform exact string matching between the system-generated text and target text, METEOR uses WordNet to match synonyms because the meanings are the same. In addition, it proposes to organize words into chunks and use this to determine how well-ordered the words are. METEOR is calculated as follows:

$$\text{P} = \frac{\text{number of unigrams matched}}{\text{number of unigrams in generated text}} \tag{16}$$

$$\text{R} = \frac{\text{number of unigrams matched}}{\text{number of unigrams in reference text}} \tag{17}$$

$$\text{Fmean} = \frac{10PR}{R + 9P} \tag{18}$$

$$\text{Penalty} = 0.5 \cdot \left(\frac{\text{number of chunks}}{\text{number of unigrams matched}} \right) \tag{19}$$

$$\text{METEOR} = \text{Fmean} \cdot (1 - \text{Penalty}) \tag{20}$$

The Fmean in Equation (20) is the harmonic mean (Equation (18)) of unigram precision (Equation (16)) and unigram recall (Equation (17)), which gives more weight on recall. Penalty (Equation (19)) is calculated based on the number of chunks, which is used to measure how well-ordered the words are.

- ROUGE$_L$ [55]: Unlike BLEU, this metric mainly focuses on the models' performance on recall. Additionally, it uses the Longest Common Subsequence (LCS) to match the system-generated text and reference text.

$$R_{lcs} = \frac{LCS(\text{reference text, generated text})}{\text{length of reference text}} \tag{21}$$

$$P_{lcs} = \frac{LCS(\text{reference text, generated text})}{\text{length of generated text}} \tag{22}$$

$$ROUGE_L = \frac{(1 + \beta^2) R_{lcs} P_{lcs}}{R_{lcs} + \beta^2 P_{lcs}} \tag{23}$$

Equation (23) shows that $ROUGE_L$ is the F-measure of the LCS precision (Equation (22)) and LCS recall (Equation (21)), which places more weight on recall since β is a large number.

- CIDEr [56]: Similar to the NIST score, this metric also focus on tokens that are more informative. Differently, it uses Term Frequency Inverse Document Frequency

(TF-IDF) to serve the purpose, as it will give more weight to infrequently occurring but informative words in the corpus. CIDEr is calculated as follows:

$$g_k(r_{ij}) = \frac{h_k(r_{ij})}{\sum\limits_{w_l \in \Omega} h_l(r_{ij})} log \left(\frac{|I|}{\sum\limits_{I_p \in I} \min(1, \sum\limits_q h_k(r_{pq}))} \right) \tag{24}$$

$$\text{CIDEr}_n(c_i, r_i) = \frac{1}{m} \sum_j \frac{g^n(c_i) \cdot g^n(r_{ij})}{||g^n(c_i)||||g^n(r_{ij})||} \tag{25}$$

Equation (25) shows that CIDEr uses a TF-IDF (Equation (24)) vector to represent the generated text and reference text, and uses cosine similarity to calculate the score. m is the number of reference texts. The first fraction in Equation (24) is the n-gram w_k's term frequency in the text. The second fraction is its inverse document frequency across the documents in the corpus. Ω is the vocabulary of all n-grams and I consists of all texts in the corpus.

- TER [50]: TER is the abbreviation of the Translation Edit Rate. It measures the quality of the system-generated text by calculating the number of edit operations to an exact match between the system-generated text and reference text. TER is calculated as follows:

$$\text{TER} = \frac{\text{number of edits}}{\text{average number of words in reference text}} \tag{26}$$

- MoverScore [51]: Unlike previous metrics that compare system-generated text and reference text only in their surface form, it uses contextualized embedding with Earth Mover's Distance to evaluate the texts in the semantic level. MoverScore is calculated as follows:

$$\begin{aligned} \text{WMD}(x^n, y^n) &= \min_{F \in R^{|x^n| \times |y^n|}} < C, F >, \\ \text{s.t.} F1 &= f_{x^n}, F^T 1 = f_{y^n} \end{aligned} \tag{27}$$

x^n and y^n represent the generated text and reference text. C is a matrix of transportation costs to transform x^n into y^n. F consists of n-gram weights. f_{x^n} is the n-gram weights for text sequence x^n. f_{y^n} is the n-gram weights for text sequence y^n. This criterion measures the semantic distance between x^n and y^n.

- BLEURT [53] and BERTScore [52]: Similar to MoverScore, they are different from those n-gram-based criteria by using a pretrained language model to measure performance in the semantic level. BERTScore is calculated as follows:

$$R_{\text{BERT}} = \frac{1}{|x|} \sum_{x_i \in x} max_{\hat{x}_j \in \hat{x}} x_i^T \hat{x}_j \tag{28}$$

$$P_{\text{BERT}} = \frac{1}{|\hat{x}|} \sum_{\hat{x}_j \in \hat{x}} max_{x_i \in x} x_i^T \hat{x}_j \tag{29}$$

$$F_{\text{BERT}} = 2 \frac{P_{\text{BERT}} \cdot R_{\text{BERT}}}{P_{\text{BERT}} + R_{\text{BERT}}} \tag{30}$$

BERTScore (Equation (30)) is the combination of precision (Equation (29)) and recall (Equation (28)) scores of generated text $\hat{x}$ and reference text x. P_{BERT} and R_{BERT} maximize the similarity score between the two texts via greedy matching.
BLEURT is calculated as follows:

$$\hat{y} = f(x, \hat{x}) = W\hat{v}_{CLS} + b \tag{31}$$

Equation (31) uses BERT [57] to model both generated text $\hat{x}$ and reference text x and uses its representation of the special token (CLS) to predict the rating of the generated text via linear transformation. W and b are trainable parameters.

5.3. Comparing Methods

We compare the following methods on data-to-text generation datasets:

- Diffuseq [26]: While some attempts are made for language modeling with a diffusion model, this is the first and latest diffusion-based sequence-to-sequence model for text generation. We ran their publicly accessible codes and the suggested hyperparameters on WebNLG, E2E, and DART datasets.
- GAS [58]: They propose to model the local and global structural information through a graph neural network and use reinforcement learning to train the model.
- LoRA [39]: LoRA stands for Low-Rank Adaptation, which inserts trainable rank decomposition matrices into each layer of the model and fixes the parameters of the pretrained language model (PLM) to reduce the cost of finetuning.
- Xie et al. [59]: they explore the pretrained language model's performance on data-to-text generation tasks.
- HierBlock [60]: Based on prefix tuning, this paper integrates hierarchical discourse information into modeling.
- Shen et al. [61]: They draw inspiration from computational pragmatics, which follows the intuitive that the "speaker should generate output text that a listener can use to identify correctly". They experiment on the E2E dataset, and we report the result in the table.
- An et al. [62]: They propose to finetune both prefixes and input data's representations so that PLM can better understand unfamiliar input data.
- Hou et al. [63]: They propose to use meta learning to adapt PLM to different tasks better.
- Finetune: This stands for finetuning the corresponding pretrained language model on the corresponding datasets.
- Adapter [64]: It introduces a trainable small-scale neural network module adapter to attach to the pretrained language model. For different tasks, different adapters can be attached to a single PLM to adapt to different tasks.
- Prefix [18]: It freezes the pretrained language model while prepending multiple trainable prefixes to the PLM to adapt it to different tasks. This can be considered our base model.
- +Diff: We combine the diffusion model with prefix tuning. This is the model described in Section 4.1.
- +Diff+LG: Additionally, we apply the look-ahead guiding loss (Section 4.2) to +Diff, in order to better supervise the denoising process of the diffusion model.
- +Diff+LG+CS: This is the full model, which combines the diffusion model (Section 4.1), look-ahead guiding loss (Section 4.2) and the planning-then-writing pipeline (Section 4.3).

5.4. Implementation Details

For all of three data-to-text generation tasks, we conducted experiments on two types of backbone pretrained language models, including GPT-2 Median and GPT-2 Large. Regardless of whether the input structured data are triple or <attribute, value> pairs, we use the rule to linearize them into a sequence that is in the form of natural language, as suggested by Li and Liang [18]. We use AdamW as the optimizer and linear learning rate scheduler as suggested by Li and Liang [18]. We select the number of epochs as 10 from {5, 10, 15}, learning rate as 0.00015 from {0.00005, 0.0001, 0.00015, 0.0002}, and prefix length as 15 from {5, 10, 15, 20}. We also tune the introduced hyperparameters in this paper. The diffusion step is set to be 20 from {10, 20, 50, 100, 200, 500}, pretrained number of epochs for the content planning stage as 4 from {3, 4, 5} and ξ as 1.0 from {0.5, 1.0, 1.5}. We did not specifically tune the batch size, but used the largest batch size 15 that can fit in the GPU. The models are trained on a single NVIDIA A100 80GB, and it takes 9 min to train an epoch on the WebNLG dataset, which has 18K training data, using GPT-2 Large. During decoding, we follow suggestions by Li and Liang [18] and use beam search with the beam size as 5.

5.5. Automatic Evaluation Results

Tables 3–5 show the automatic evaluation results on the three data-to-text generation datasets. We observe the following facts:

- Our models outperform the methods of finetune, adapter, and prefix tuning on the WebNLG dataset's whole test set (A category), with respect to both GPT-2 Medium or GPT-2 Large as the backbone. For example, our full model +Diff+LG+CS outperforms the prefix tuning by 1.31% on BLEU with GPT-2 Medium. Additionally, our full model outperforms the prefix tuning by 2.19% on BLEU with GPT-2 Large. In addition, our full model (+Diff+LG+CS) on GPT-2 Large outperforms Diffuseq, a Sequence-to-Sequence diffusion-based text generation method, and other competitive baselines as listed in the part of Other Methods. The results above show the effectiveness of our proposed model.
- Our models also outperform finetuning, adapter, and prefix tuning on GPT-2 Medium by 1.69% in terms of BLEU, compared with prefix tuning and 1.66% in terms of BLEU on GPT-2 Large on the DART dataset. The comparison with other methods also indicates the effectiveness of our model on the DART dataset.
- Our models show similar patterns as described above on the E2E dataset. The full model improves prefix tuning by 0.88% in terms of BLEU on GPT-2 Medium and 1.14% on GPT-2 Large. They also outperform other methods on most metrics.

Since WebNLG provides unique and comprehensive evaluation metrics that show the models' ability on both seen and unseen domains, we find that the finetuning method excels in the seen domains, as it can better use learned knowledge on those domains. However, it comes at the price of the generalization ability of the model, as its performance on unseen domains lags behind prefix-tuning-based methods, especially on BLEU and TER. This shows the great potential of generalization for the prefix-tuning method. Additionally, our model can push the generalization ability even further, as indicated by the BLEU score on unseen domains in WebNLG.

Table 3. Automatic evaluation results (higher is better, except for TER) for data-to-text generation on WebNLG dataset. The best scores are boldfaced for pretrained language model (PLM) GPT-2$_{\text{MEDIUM}}$ and GPT-2$_{\text{LARGE}}$, respectively. In the header, S refers to the seen category in WebNLG's test set, which means the domain of this subset of test set is seen during training. U refers to the unseen category, which means the domain of this subset of test set is never seen during training. A is the average performance across seen and unseen categories. +Diff+LG+CS is our proposed full model with the three modules described in Section 4. +Diff and +Diff+LG are two ablations. +Diff only uses the module in Section 4.1 and +Diff+LG use both modules in Sections 4.1 and 4.2. Our full model improves the prefix tuning method by 2.19% in terms of BLEU metric in A category (58.61 vs. 57.35).

		BLEU			**MET**			**TER ↓**	
	S	**U**	**A**	**S**	**U**	**A**	**S**	**U**	**A**
					Other Methods				
DiffuSeq	42.74	7.80	28.07	36.52	17.29	27.36	47.62	77.15	61.16
GAS [58]	57.79	26.55	44.00	41.00	26.00	34.00	41.00	66.00	53.00
LoRA [39]	64.00	48.40	57.00	45.00	39.00	42.00	32.00	45.00	38.00
					GPT-2$_{\text{MEDIUM}}$				
Finetune	**64.13**	35.34	50.57	**45.67**	34.51	40.33	33.19	64.73	47.65
Adapter	54.71	43.66	49.72	39.64	35.38	37.67	39.85	47.33	43.28
Prefix	62.29	47.23	55.49	44.38	38.22	41.51	33.71	46.04	39.36
+Diff	62.48	47.30	55.63	44.22	38.23	41.43	33.84	46.00	39.41
+Diff+LG	62.87	47.43	55.91	44.49	38.09	41.51	**33.18**	**45.41**	**38.78**
+Diff+LG+CS	63.06	**47.86**	**56.22**	44.77	**38.69**	**41.93**	33.74	46.29	39.50
					GPT-2$_{\text{LARGE}}$				
Finetune	63.60	45.46	55.78	44.80	38.49	41.83	34.10	51.35	42.01
Adapter	62.04	49.39	56.30	43.83	38.48	41.35	34.23	44.35	38.87
Prefix	64.43	48.72	57.35	45.61	39.20	42.61	32.70	45.82	38.72
+Diff	63.63	50.11	57.52	44.96	39.46	42.39	33.35	44.75	38.58
+Diff+LG	**65.19**	48.87	57.84	**45.86**	39.12	42.71	**32.24**	46.23	38.65
+Diff+LG+CS	65.18	**50.58**	**58.61**	45.55	**39.78**	**42.85**	32.45	**44.01**	**37.75**

Table 4. Automatic evaluation results (higher is better, except for TER) for data-to-text generation on the DART dataset. The best scores are boldfaced for both GPT-2$_{\text{MEDIUM}}$ and GPT-2$_{\text{LARGE}}$, respectively. Our model improves the prefix tuning method by 1.66% in terms of the BLEU metric (47.60 vs. 46.82).

	BLEU	MET	TER ↓	MoverScore	BERTScore	BLEURT
			Other Methods			
DiffuSeq	12.63	31.77	60.53	62.84	89.01	27.62
Xie et al. [59]	46.89	55.76	60.97	-	95.00	30.00
GAS [58]	39.87	32.00	57.00	-	-	-
HierBlock [60]	46.60	39.00	45.00	54.00	95.00	-
LoRA [39]	47.50	39.00	45.00	-	-	-
			GPT-2$_{\text{MEDIUM}}$			
Finetune	45.98	38.57	45.70	67.86	94.82	39.07
Adapter	42.77	36.59	47.96	66.57	94.48	34.23
Prefix	45.94	38.49	45.49	67.94	94.90	39.80
+Diff	46.06	**38.79**	46.01	**68.10**	94.88	**40.69**
+Diff+LG	46.39	38.75	45.47	**68.10**	94.91	40.47
+Diff+LG+CS	**46.72**	38.70	**45.41**	68.06	**94.95**	40.34
			GPT-2$_{\text{LARGE}}$			
Finetune	46.90	39.01	45.09	68.23	94.95	40.08
Adapter	46.24	38.41	45.56	67.89	94.88	39.22
Prefix	46.82	38.90	45.06	68.33	94.99	41.17
+Diff	47.15	38.96	44.73	68.31	95.01	41.26
+Diff+LG	47.43	38.96	**44.70**	68.35	95.03	41.23
+Diff+LG+CS	**47.60**	**39.21**	**44.70**	**68.45**	**95.05**	**41.60**

Table 5. Automatic evaluation results (higher is better, except for TER) for data-to-text generation on the E2E dataset. The best score are boldfaced for both GPT-2$_{\text{MEDIUM}}$ and GPT-2$_{\text{LARGE}}$, respectively. Our model improves the prefix tuning by 1.14% in terms of BLEU (70.89 v.s. 70.09).

	BLEU	NIST	MET	ROUGE-L	CIDEr
			Other Methods		
DiffuSeq	48.90	7.37	39.34	59.15	1.53
Shen et al. [61]	68.60	8.73	45.25	70.82	2.37
An et al. [62]	68.70	8.74	46.10	70.70	2.42
HierBlock [60]	67.20	8.70	45.10	69.10	2.35
LoRA [39]	70.40	8.89	46.80	72.00	2.47
Hou et al. [63]	69.70	8.78	46.90	72.10	2.51
			GPT-2$_{\text{MEDIUM}}$		
Finetune	68.37	8.71	45.93	70.89	2.41
Adapter	66.24	8.53	43.59	69.21	2.26
Prefix	69.75	8.79	46.39	71.54	2.51
+Diff	70.03	8.81	**46.57**	71.64	2.51
+Diff+LG	69.83	8.79	46.21	71.38	2.50
+Diff+LG+CS	**70.37**	**8.86**	46.42	**71.77**	**2.52**
			GPT-2$_{\text{LARGE}}$		
Finetune	68.98	8.77	45.91	71.36	2.42
Adapter	68.73	8.68	46.28	71.01	2.50
Prefix	70.09	8.82	46.33	72.13	2.48
+Diff	70.34	8.85	46.38	71.76	**2.52**
+Diff+LG	70.43	8.86	46.24	71.83	2.48
+Diff+LG+CS	**70.89**	**8.93**	**46.51**	**72.14**	2.51

5.6. Ablation Study

We also include the ablation study results in Tables 3–5. Based on the prefix tuning baseline, we progressively add the proposed modules and observe each module's contribution to the improvement, where +Diff only employs DiffuD2T in Section 4.1, +Diff+LG employs both DiffuD2T and look-ahead guiding loss (Section 4.2) and +Diff+LG+CS refers to the full model that additionally includes the planning-then-writing pipeline (Section 4.3). As shown in the tables, our model peaks the performance when the diffusion model, look-ahead guiding loss, and the planning-then-writing training pipeline are combined in most cases with few exceptions. This demonstrates the effectiveness of all our proposed modules.

5.7. Human Evaluation

In order to provide a more comprehensive analysis of the models' performance, we employ three graduates who are proficient in English to conduct the human evaluation, comparing the performance of Finetune, Prefix tuning, and our full model +Diff+LG+CS, trained with GPT-2 Large, on all three datasets. For each dataset and each model, we sample 50 examples from the test set, arrange the generated text from three models into three pairs, and ask the human raters to determine which one in the pairs is better. Following Dušek et al. [65], we ask raters to consider two aspects: quality and naturalness. Quality focuses on whether the text faithfully describes information in the structured data, and naturalness focuses on whether the text is fluent. The results in the table are calculated as the percentage of times a model is deemed better minus the percentage of times a model is deemed worse. The results range from −100% to 100%, and the higher the results, the better quality of the text. We also use Kappa [66] to analyze rater agreement. The results are illustrated in Table 6's caption. As shown in Table 6, in terms of both the faithfulness of the text (Quality) and the fluency of the text (Naturalness), prefix tuning performs better than finetuning using the GPT-2 Large language model on all three datasets. Additionally, our proposed model +Diff+LG+CS is better than the prefix tuning across all datasets and human evaluation metrics.

Table 6. Human evaluation results on WebNLG, DART, and E2E datasets. Quality focuses on whether the text faithfully describes information in the structured data, and naturalness focuses on whether the text is fluent. The results in the table are calculated as the percentage of times a model is deemed better minus the percentage of times a model is deemed worse. We utilize Kappa [66] to analyze the rater agreement. In terms of quality, the kappas are 0.36 for the WebNLG dataset, 0.45 for the DART dataset, and 0.55 for the E2E dataset. In terms of naturalness, the kappas are 0.46 for the WebNLG dataset, 0.52 for the DART dataset, and 0.44 for the E2E dataset. The best results in different datasets are in bold.

Dataset	Model	Quality	Naturalness
WebNLG	Finetune	−6.11	−19.22
	Prefix	9.67	−1.22
	+Diff+LG+CS	**28.22**	**32.00**
DART	Finetune	−0.67	−18.67
	Prefix	15.67	1.44
	+Diff+LG+CS	**33.00**	**32.11**
E2E	Finetune	1.33	−17.22
	Prefix	7.67	-3.33
	+Diff+LG+CS	**39.44**	**37.67**

6. Case Study

6.1. Diffusion Steps

We present generated texts using the iterative refinement process of the diffusion model with different steps (Table 7). The settings of this case study are listed below:

- The structured data and corresponding reference text are sampled from the test set of WebNLG.
- Given the structured data, we use our proposed full model +Diff+LG+CS model with GPT-2 Large as its pretrained language model backbone to generate text. We explore two settings of diffusion steps: 10 denoising steps and 20 denoising steps (Section 4.1).

These examples show that the iterative refinement process of the diffusion model can perfect the representation of the prefixes step-by-step and improve the quality of text. For example, in the first example, the generated text via 10 diffusion steps missed important information such as "He was a crew member of Apollo 14", while more diffusion steps mitigate this problem. Additionally, in the second example, the generated text produced using 10 diffusion steps misunderstood the relationship between “Ticino” and “Mendrisio”, and put “Ticino” before “Mendrisio” while Mendrisio is a municipality in the district of Mendrisio in the canton of Ticino in Switzerland.

6.2. Content Planning

We presented generated texts from the content planning stage (Table 8) to provide an intuitive illustration of our proposed planning-then-writing pipeline (Section 4.3). The settings of this case study are listed below:

- The data and corresponding reference text are sampled from WebNLG’s test set.
- Content plans are generated by our proposed full model +Diff+LG+CS model with GPT-2 Large as its pretrained language model backbone.
- Extracted plans are automatically extracted from reference text (Section 4.3) and used to train the content planner.

As for the content planning stage’s performance, we have the following observations:

- As shown in all three examples, our model, trained in the content planning stage, obtains good coverage performance, that is, the information in the data is covered in the content plans, by comparing the generated plan and extracted plan.
- However, the overall order of the data in the plans can be different from the one in the reference text, indicating a different style of writing compared to the reference. For instance, in the second example, the generated plan puts “Costa Crociere” and “Genoa” at the end of the plan while the extracted plan from the reference text puts them at the start of it, which means the model tends to first describe “The operator of AIDAstella” before mentioning “Costa Crociere” in the next surface realization stage, as illustrated in Table 9. However, our model may make mistakes during content planning. In the first example, the generated plan puts “Amar Osim” at the end of the plan, which makes it less natural, while the extracted plan and reference text put it in the middle of the text. However, in the planning-then-writing pipeline, models for surface realization can mitigate this, as illustrated in Table 9.

Table 7. Generated examples to examine the effect of different diffusion steps on our proposed model (+Diff+LG+CS). The data and reference text are from the WebNLG test set. We present the generation results of utilizing the iterative refinement process of the diffusion model for 10 steps and for 20 steps. Texts are generated by the +Diff+LG+CS model with GPT-2 Large as its pretrained language model backbone.

Data	Diffusion Steps: 10	Diffusion Steps: 20	Reference Text
(Apollo 14, was a crew member of, Alan Shepard), (Department of Commerce Gold Medal, higher, Distinguished Service Medal (United States Navy)), (California, deathPlace, Alan Shepard), (New Hampshire, birthPlace, Alan Shepard), (NASA, operator, Apollo 14), (Distinguished Service Medal (United States Navy), awards, Alan Shepard)	Alan Shepard was born in New Hampshire and died in California. He was awarded the Distinguished Service Medal by the United States Navy, which ranks higher than the Department of Commerce Gold Medal.	Alan Shepard was born in New Hampshire and died in California. He was a crew member of Apollo 14 operated by NASA and was awarded the Distinguished Service Medal by the United States Navy, which is higher than the Department of Commerce Gold Medal.	Alan Shepard was born in New Hampshire and died in California. He was a crew member of Apollo 14 which is operated by NASA. He was awarded the Distinguished Service Medal in the US Navy, which is higher than the Dept of Commerce Gold Medal.
(Switzerland, country, Accademia di Architettura di Mendrisio), (Mario Botta, dean, Accademia di Architettura di Mendrisio), (Mendrisio, city, Accademia di Architettura di Mendrisio), (1996, established, Accademia di Architettura di Mendrisio), (100, academicStaffSize, Accademia di Architettura di Mendrisio), (Ticino, location, Accademia di Architettura di Mendrisio)	The Accademia di Architettura di Mendrisio is located in Ticino, Mendrisio, Switzerland. It was established in 1996 and has 100 academic staff. Its dean is Mario Botta.	The Accademia di Architettura di Mendrisio in Ticino, Switzerland was established in 1996. Its dean is Mario Botta and it has 100 academic staff.	Accademia di Architettura di Mendrisio in Mendrisio, Switzerland has 100 employees, was established in 1996 and is overseen by Dean Mario Botta. The school’s location is Ticino.
(2006-12-31, epoch, 110 Lydia), (6.7 (kilograms), mass, 110 Lydia), (377,016,000,000.0, periapsis, 110 Lydia)	110 Lydia has a mass of 6.7 (kilograms) and a periapsis of 377,016,000,000.0. It was published in 2006.	110 Lydia, which has a mass of 6.7 kg and periapsis of 377,016,000,000.0, was published in 2006.	110 Lydia’s epoch is 31 December 2006, it has a mass of 6.7 kg and a periapsis of 377,016,000,000.0.

Table 8. Generated examples during the content planning stage (+Diff+LG+CS) in the planning-then-writing pipeline (Section 4.3). The data, extracted plan, and reference text are from the WebNLG test set, and the generated plan is from the +Diff+LG+CS model with GPT-2 Large as its pretrained language model backbone.

Data	Generated Plan	Extracted Plan	Reference Text
(Al Kharaitiyat SC, club, Alaa Abdul-Zahra), (Al Khor, ground, Al Kharaitiyat SC), (Shabab Al-Ordon Club, club, Alaa Abdul-Zahra) (Amar Osim, manager, Al Kharaitiyat SC)	Content plans are: Alaa Abdul Zahra; Al Kharaitiyat SC; Al Khor; Shabab Al Ordon Club; Amar Osim.	Content plans are: Alaa Abdul Zahra; Al Kharaitiyat SC; Al Khor; Amar Osim; Shabab Al Ordon club; Alaa Abdul Zahra.	Alaa Abdul Zahra plays for Al Kharaitiyat SC which is located at Al Khor and managed by Amar Osim. The Shabab Al-Ordon club is associated with Alaa Abdul-Zahra.
(Genoa, location, Costa Crociere), (AIDA Cruises, operator, AIDAstella), (Costa Crociere, owner, AIDAstella)	Content plans are: AIDAstella; AIDA Cruises; Costa Crociere; Genoa.	Content plans are: Costa Crociere; AIDAstella; Genoa; AIDAstella; AIDA Cruises.	Costa Crociere is the owner of the AIDAstella and are based in Genoa. The operator of AIDAstella is AIDA Cruises.
(Euro, currency, Netherlands), (Stellendam, birthPlace, Ab Klink), (Netherlands, nationality, Ab Klink)	Content plans are: Ab Klink; Stellendam; Netherlands; Euro.	Content plans are: Ab Klink; Stellendam; Netherlands; euro.	Ab Klink was born in Stellendam in the Netherlands, where the national currency is the euro.

Table 9. Examples of models' generated text on the test set of WebNLG. Following the setting in Li and Liang [18], the 6 examples are from the unseen categories, that is, the domain of these examples is not seen during training, which shows the generalization ability of the models. +Diff+LG+CS is our model. Prefix and Finetune are two competitive methods, and the reference is the gold text. All three models use GPT-2 Large as their pretrained language model backbone. Text that contradicts the structured data is marked in **red**.

Data Source or Model Name	Content
Source [Unseen, Athelete]	(Al Kharaitiyat SC, club, Alaa Abdul-Zahra), (Al Khor, ground, Al Kharaitiyat SC), (Shabab Al-Ordon Club, club, Alaa Abdul-Zahra) (Amar Osim, manager, Al Kharaitiyat SC)
+Diff+LG+CS	Alaa Abdul-Zahra plays for Al Kharaitiyat SC, which is managed by Amar Osim. He also plays for Shabab Al-Ordon Club.
Prefix	Al Kharaitiyat SC are managed by Amar Osim and play at their ground at Al Khor. Al Kharaitiyat SC are also the club for which Alaa Abdul-Zahra is a player.
Finetune	Alaa Abdul-Zahra plays for Al-Kharaitiyat SC and Shabab Al-Ordon Club. He also plays for Al-Khor and **manages** Al-Kharaitiyat SC.
Reference	Alaa Abdul Zahra plays for Al Kharaitiyat SC which is located at Al Khor and managed by Amar Osim. The Shabab Al-Ordon club is associated with Alaa Abdul-Zahra.
Source [Unseen, Transportation]	(Genoa, location, Costa Crociere), (AIDA Cruises, operator, AIDAstella), (Costa Crociere, owner, AIDAstella)
+Diff+LG+CS	AIDAstella is operated by AIDA Cruises and owned by Costa Crociere in Genoa.
Prefix	AIDAstella is operated by Aida Cruises and is owned by the **Costa Rican tourist resort** of Genoa.
Finetune	AIDAstella, operated by AIDA-Cruises, is located in Genoa and is owned by the **Costa Rican government**.
Reference	Costa Crociere is the owner of the AIDAstella and are based in Genoa. The operator of AIDAstella is AIDA Cruises.
Source [Unseen, Politician]	(Euro, currency, Netherlands), (Stellendam, birthPlace, Ab Klink), (Netherlands, nationality, Ab Klink)
+Diff+LG+CS	Ab Klink was born in Stellendam, the Netherlands, where the currency is the Euro.
Prefix	Ab Klink was born in Stellendam and is a national of the Netherlands where the currency is the Euro.
Finetune	Ab Klink is a national of the Netherlands where the currency is the Euro. He was born in Stellendam.
Reference	Ab Klink was born in Stellendam in the Netherlands, where the national currency is the euro.
Source [Unseen, Politician]	(Robert E, Lee, commander, Battle of Salem Church), (American Civil War, isPartOfMilitaryConflict, Battle of Salem Church), (Battle of Salem Church, battles, Aaron S. Daggett)
+Diff+LG+CS	Robert E. Lee is the commander of the Battle of Salem Church which is part of the American Civil War. The Battle of Salem Church is a military conflict.
Prefix	Robert E. Lee was the commander of the Battle of Salem Church which was part of the military conflict in the American Civil war.
Finetune	The Battle of Salem Church is part of the American Civil War and was commanded by Robert E. Lee.
Reference	Robert E Lee was a commander in the Battle of Salem Church, which was one of the military conflicts in the American Civil War. Aaron S Daggett fought in the same battle.

Table 9. *Cont.*

Data Source or Model Name	Content
Source [Unseen, Artist]	(Christian alternative rock, musicSubgenre, Alternative rock), (Alternative rock, genre, Andrew White (musician))
+Diff+LG+CS	Christian alternative rock is the genre of music Andrew White (musician) plays.
Prefix	Andrew White is a Christian alternative rock musician.
Finetune	Andrew White, a Christian alternative rocker, performs.
Reference	The musician Andrew White's genre is alternative rock, the genre which has the sub genre Christian alternative rock.
Source [Unseen, Artist]	(Hip hop music, genre, Allen Forrest), (solo singer, background, Allen Forrest)
+Diff+LG+CS	Allen Forrest is a solo singer who was born in the genre of Hip Hop music.
Prefix	Allen Forrest is a solo singer.
Finetune	**Born in**
Reference	Allen Forrest is a solo singer whose genre is Hip Hop music.

6.3. Surface Realization

Following the case study setting in Li and Liang [18], the settings of this case study are listed below:

- We sample six structured data and reference text from the test set of WebNLG's unseen categories, that is the domains of these examples are not seen during training, to demonstrate the generalization ability of the models. We compare our proposed full model +Diff+LG+CS with two competitive methods: Prefix and Finetune. All three models use GPT-2 Large as their pretrained language model backbone. The results are reported in Table 9.
- We sample three structured data and reference text from the test set of E2E and three from the test set of DART. We compare our proposed full model +Diff+LG+CS with the baseline model Prefix. Both models use GPT-2 Large as its pretrained language model backbone. Results are reported in Table 10.

Table 10. Examples of the models' generated text on the test set of the E2E and DART datasets. +Diff+LG+CS is our model. Prefix is the competitive baseline method. Both models use GPT-2 Large as its pretrained language model backbone. The reference is the gold text.

Data Source or Model Name	Content
Source [E2E]	(name, Cocum), (Type, coffee shop), (customer rating, high), (near, Burger King)
+Diff+LG+CS	Cocum is a highly rated coffee shop near Burger King.
Prefix	Cocum is a coffee shop near Burger King with a high customer rating.
Reference	Near Burger King there is a highly rated coffee shop named Cocum.
Source [E2E]	(name, The Mill), (Type, restaurant), (food, English), (price, less than £ 20), (customer rating, low), (area, city centre), (family friendly, no), (near, Café Rouge)
+Diff+LG+CS	The Mill is a restaurant providing English food in the less than £ 20 price range. It is located in the city centre near Café Rouge. It has a low customer rating and is not family - friendly.
Prefix	The Mill is a restaurant providing English food in the less than £ 20 price range. It is located in the city centre near Café Rouge. Its customer rating is low.
Reference	The Mill is restaurant in the city centre, near Café Rouge, serving low-priced English food. It is has a low customer rating and is not family-friendly.
Source [E2E]	(name, The Plough), (Type, pub), (food, Chinese), (price, cheap), (area, riverside), (family friendly, yes), (near, Raja Indian Cuisine)
+Diff+LG+CS	The Plough is a family friendly Chinese pub in the riverside area near Raja Indian Cuisine with a cheap price range.
Prefix	The Plough is a pub providing Chinese food in the cheap price range. It is located in the riverside. It is near Raja Indian Cuisine.
Reference	The Plough is a Chinese pub and pub in the riverside area near Raja Indian Cuisine. It is family friendly and has cheap pricing.

Table 10. *Cont.*

Data Source or Model Name	Content
Source [DART]	(Mark Rutte, leader_name, Netherlands)
+Diff+LG+CS Prefix Reference	The leader of the Netherlands is Mark Rutte. Mark Rutte is the leader of the Netherlands. The leader of the Netherlands is Mark Rutte.
Source [DART]	(Michigan, country, United States)
+Diff+LG+CS Prefix Reference	Michigan is in the United States. Michigan is located within the United States. Michigan is in the United states.
Source [DART]	(Alfa Romeo 164, assembly, Arese)
+Diff+LG+CS Prefix Reference	Alfa Romeo 164 was assembled by Arese. Alfa Romeo 164 is assembled arese. The Alfa Romeo 164 was assembled in Arese.

As for the cases, we have the following observations:

- In unseen domains (Table 9), both the prefix-tuning method and finetuning method tend to omit some of the information in the data, as the pattern may not be seen during training. However, our model (+Diff+LG+CS) can sometimes generalize better. For instance, in the sixth example, prefix tuning omitted the genre of the music and finetuning failed to generate fluent text, while our model can include both pieces of information in the structured data.
- Additionally, our model can sometimes generate more faithful content when the domain is unseen. In the second example, both the prefix-tuning and finetuning methods failed to understand what the "owner" in the structured data meant and generated the incorrect owner in this case. Our model can generate it correctly.
- As for generation results on E2E and DART datasets (Table 10), our model can generate more faithful text with higher coverage, that is, cover more information in data, as shown in the second example. In the second example, our model correctly describes "not family-friendly", while prefix tuning omitted this information. Additionally, the third example shows that our model can generate more fluent text than prefix tuning.

7. Conclusions

Data-to-text generation has become an important research task in natural language processing. In this paper, we explore how to introduce an iterative refinement process for data-to-text generation. We propose three modules: (1) we use the diffusion model to improve data-to-text generation with the iterative refinement process; (2) we propose a look-ahead guiding loss to supervise the iterative refinement process; and (3) we extract content plans from the reference text and propose a planning-then-writing pipeline that can give the model content planning ability. We conducted experiments on three data-to-text generation benchmarks. The automatic evaluation results show that our model can outperform the prefix tuning model in terms of the BLEU (BiLingual Evaluation Understudy) metric by 2.19% on the WebNLG dataset with GPT-2 Large as its backbone. Human evaluations and analyses show that our model can generate high-quality and more natural text across different datasets.

However, this work also has its limitation that we extract the content plans from reference text for our planning-then-writing pipeline heuristically. The extracted plan may miss some information in the text, creating noise in the training process. In the future, we will explore more sophisticated extraction methods to improve the quality of the extracted plans for training.

Author Contributions: Conceptualization, H.G., X.F. and B.Q.; formal analysis, H.G.; funding acquisition, B.Q.; investigation, H.G.; methodology, H.G.; project administration, X.F. and B.Q.; resources, H.G.; software, H.G.; supervision, X.F. and B.Q.; validation, H.G., X.F. and B.Q.; visualization, H.G.; writing—original draft, H.G.; writing—review and editing, H.G., X.F. and B.Q. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the National Key R&D Program of China via grant 2020AAA0106502, National Natural Science Foundation of China (NSFC) via grant 62276078, and the Province Key R&D Program of Heilongjiang via grant 2022ZX01A32.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All data included in this study are available upon request by contact with the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Gatt, A.; Portet, F.; Reiter, E.; Hunter, J.; Mahamood, S.; Moncur, W.; Sripada, S. From data to text in the neonatal intensive care unit: Using NLG technology for decision support and information management. *Ai Commun.* **2009**, *22*, 153–186. [CrossRef]
2. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to sequence learning with neural networks. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 3104–3112.
3. Luong, T.; Pham, H.; Manning, C.D. Effective Approaches to Attention-based Neural Machine Translation. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015; pp. 1412–1421.
4. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.u.; Polosukhin, I. Attention is All you Need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4 December 2017; pp. 6000–6010.
5. Novikova, J.; Dušek, O.; Rieser, V. The E2E Dataset: New Challenges For End-to-End Generation. In Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue, Saarbrücken, Germany, 15–17 August 2017; pp. 201–206.
6. Gardent, C.; Shimorina, A.; Narayan, S.; Perez-Beltrachini, L. The WebNLG Challenge: Generating Text from RDF Data. In Proceedings of the 10th International Conference on Natural Language Generation, Santiago de Compostela, Spain, 4–7 September 2017; pp. 124–133.
7. Nan, L.; Radev, D.; Zhang, R.; Rau, A.; Sivaprasad, A.; Hsieh, C.; Tang, X.; Vyas, A.; Verma, N.; Krishna, P.; et al. DART: Open-Domain Structured Data Record to Text Generation. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Online, 6–11 June 2021; pp. 432–447.
8. Puduppully, R.; Dong, L.; Lapata, M. Data-to-Text Generation with Content Selection and Planning. In Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January 2019; pp. 6908–6915.
9. Puduppully, R.; Dong, L.; Lapata, M. Data-to-text Generation with Entity Modeling. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July 2019; pp. 2023–2035.
10. Chen, K.; Li, F.; Hu, B.; Peng, W.; Chen, Q.; Yu, H.; Xiang, Y. Neural data-to-text generation with dynamic content planning. *Knowl.-Based Syst.* **2021**, *215*, 106610. [CrossRef]
11. Puduppully, R.; Lapata, M. Data-to-text Generation with Macro Planning. *Trans. Assoc. Comput. Linguist.* **2021**, *9*, 510–527. [CrossRef]
12. Nie, F.; Wang, J.; Yao, J.G.; Pan, R.; Lin, C.Y. Operation-guided Neural Networks for High Fidelity Data-To-Text Generation. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October 2018; pp. 3879–3889.
13. Zhao, C.; Walker, M.; Chaturvedi, S. Bridging the Structural Gap Between Encoding and Decoding for Data-To-Text Generation. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; pp. 2481–2491.
14. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language Models are Unsupervised Multitask Learners. *OpenAI Blog* **2019**, *1*, 9.
15. Chen, W.; Su, Y.; Yan, X.; Wang, W.Y. KGPT: Knowledge-Grounded Pre-Training for Data-to-Text Generation. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Online, 16–20 November 2020; pp. 8635–8648.
16. Chen, Z.; Eavani, H.; Chen, W.; Liu, Y.; Wang, W.Y. Few-Shot NLG with Pre-Trained Language Model. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; pp. 183–190.
17. Chang, E.; Shen, X.; Zhu, D.; Demberg, V.; Su, H. Neural Data-to-Text Generation with LM-based Text Augmentation. In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, Online, 19–23 April 2021; pp. 758–768.

18. Li, X.L.; Liang, P. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Online, 1–6 August 2021; pp. 4582–4597.
19. Sohl-Dickstein, J.; Weiss, E.; Maheswaranathan, N.; Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In Proceedings of the International Conference on Machine Learning, Lille, France, 7–9 July 2015; pp. 2256–2265.
20. Ho, J.; Jain, A.; Abbeel, P. Denoising diffusion probabilistic models. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 7–12 December 2020; pp. 6840–6851.
21. Nichol, A.Q.; Dhariwal, P.; Ramesh, A.; Shyam, P.; Mishkin, P.; Mcgrew, B.; Sutskever, I.; Chen, M. GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models. In Proceedings of the International Conference on Machine Learning, Baltimore, MA, USA, 17–23 July 2022; pp. 16784–16804.
22. Dhariwal, P.; Nichol, A. Diffusion models beat gans on image synthesis. In Proceedings of the Advances in Neural Information Processing Systems, Online, 6–14 December 2021; pp. 8780–8794.
23. Kong, Z.; Ping, W.; Huang, J.; Zhao, K.; Catanzaro, B. DiffWave: A Versatile Diffusion Model for Audio Synthesis. In Proceedings of the International Conference on Learning Representations, Vienna, Austria, 3–7 May 2021; pp. 1–17.
24. Yang, D.; Yu, J.; Wang, H.; Wang, W.; Weng, C.; Zou, Y.; Yu, D. Diffsound: Discrete diffusion model for text-to-sound generation. *arXiv* **2022**, arXiv:2207.09983.
25. Li, X.; Thickstun, J.; Gulrajani, I.; Liang, P.S.; Hashimoto, T.B. Diffusion-LM Improves Controllable Text Generation. In Proceedings of the Advances in Neural Information Processing Systems, New Orleans, LA, USA, 28 November 2022; pp. 4328–4343.
26. Gong, S.; Li, M.; Feng, J.; Wu, Z.; Kong, L. DiffuSeq: Sequence to Sequence Text Generation with Diffusion Models. In Proceedings of the International Conference on Learning Representations, Kigali, Rwanda, 1–5 May 2023; pp. 1–20.
27. Strudel, R.; Tallec, C.; Altché, F.; Du, Y.; Ganin, Y.; Mensch, A.; Grathwohl, W.; Savinov, N.; Dieleman, S.; Sifre, L.; et al. Self-conditioned Embedding Diffusion for Text Generation. *arXiv* **2022**, arXiv:2211.04236.
28. Wiseman, S.; Shieber, S.; Rush, A. Challenges in Data-to-Document Generation. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, 7–11 September 2017; pp. 2253–2263.
29. Reiter, E.; Dale, R. *Building Natural Language Generation Systems*; Cambridge University Press: Cambridge, UK, 2000.
30. Vinyals, O.; Fortunato, M.; Jaitly, N. Pointer Networks. In Proceedings of the Advances in Neural Information Processing Systems, Palais des Congrès de Montréal, Montréal, QC, Canada, 7 December 2015; pp. 2692–2700.
31. Schlichtkrull, M.; Kipf, T.N.; Bloem, P.; van den Berg, R.; Titov, I.; Welling, M. Modeling Relational Data with Graph Convolutional Networks. In Proceedings of the The Semantic Web, Cham, Switzerland, 3 June 2018; pp. 593–607.
32. Chen, W.; Chen, J.; Su, Y.; Chen, Z.; Wang, W.Y. Logical Natural Language Generation from Open-Domain Tables. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; pp. 7929–7942.
33. Chen, Z.; Chen, W.; Zha, H.; Zhou, X.; Zhang, Y.; Sundaresan, S.; Wang, W.Y. Logic2Text: High-Fidelity Natural Language Generation from Logical Forms. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2020, Online, 16–20 November 2020; pp. 2096–2111.
34. Zhang, N.; Ye, H.; Yang, J.; Deng, S.; Tan, C.; Chen, M.; Huang, S.; Huang, F.; Chen, H. LOGEN: Few-shot Logical Knowledge-Conditioned Text Generation with Self-training. *arXiv* **2021**, arXiv:2112.01404.
35. Kasner, Z.; Dusek, O. Neural Pipeline for Zero-Shot Data-to-Text Generation. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Dublin, Ireland, 22–27 May 2022; pp. 3914–3932.
36. Clive, J.; Cao, K.; Rei, M. Control Prefixes for Parameter-Efficient Text Generation. In Proceedings of the 2nd Workshop on Natural Language Generation, Evaluation, and Metrics (GEM), Abu Dhabi, United Arab Emirates, 7 December 2022; pp. 363–382.
37. Chen, Y.; Hazarika, D.; Namazifar, M.; Liu, Y.; Jin, D.; Hakkani-Tur, D. Inducer-tuning: Connecting Prefix-tuning and Adapter-tuning. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, Abu Dhabi, United Arab Emirates, 7–11 December 2022; pp. 793–808.
38. Lebret, R.; Grangier, D.; Auli, M. 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, Austin, TX, USA, 1–5 November 2016; pp. 1203–1213.
39. Hu, E.J.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W. LoRA: Low-Rank Adaptation of Large Language Models. In Proceedings of the International Conference on Learning Representations, Virtual, 25–29 April 2022; pp. 1–13.
40. Gao, Z.; Guo, J.; Tan, X.; Zhu, Y.; Zhang, F.; Bian, J.; Xu, L. Difformer: Empowering Diffusion Model on Embedding Space for Text Generation. *arXiv* **2022**, arXiv:2212.09412.
41. Ito, K.; Johnson, L. The Lj Speech Dataset 2017. Available online: https://keithito.com/LJ-Speech-Dataset (accessed on 1 February 2023).
42. Warden, P. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv* **2018**, arXiv:1804.03209.
43. Bhagavatula, C.; Le Bras, R.; Malaviya, C.; Sakaguchi, K.; Holtzman, A.; Rashkin, H.; Downey, D.; Yih, W.T.; Choi, Y. Abductive Commonsense Reasoning. In Proceedings of the International Conference on Learning Representations, Addis Ababa, Ethiopia, 26–30 April 2020.
44. Jiang, C.; Maddela, M.; Lan, W.; Zhong, Y.; Xu, W. Neural CRF Model for Sentence Alignment in Text Simplification. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; pp. 7943–7960.

45. Dhingra, B.; Mazaitis, K.; Cohen, W.W. Quasar: Datasets for question answering by search and reading. *arXiv* **2017**, arXiv:1707.03904.
46. Gu, J.; Bradbury, J.; Xiong, C.; Li, V.O.; Socher, R. Non-Autoregressive Neural Machine Translation. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April 2018.
47. Rush, A.M.; Chopra, S.; Weston, J. A Neural Attention Model for Abstractive Sentence Summarization. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015; pp. 379–389.
48. Papineni, K.; Roukos, S.; Ward, T.; Zhu, W.J. BLEU: A method for automatic evaluation of machine translation. In Proceedings of the Annual meeting on Association for Computational Linguistics, Philadelphia, PA, USA, 6 July 2002; pp. 311–318.
49. Banerjee, S.; Lavie, A. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization, Ann Arbor, MI, USA, 23 June 2005; pp. 65–72.
50. Snover, M.; Dorr, B.; Schwartz, R.; Micciulla, L.; Makhoul, J. A Study of Translation Edit Rate with Targeted Human Annotation. In Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers, Cambridge, MA, USA, 8–12 August 2006; pp. 223–231.
51. Zhao, W.; Peyrard, M.; Liu, F.; Gao, Y.; Meyer, C.M.; Eger, S. MoverScore: Text Generation Evaluating with Contextualized Embeddings and Earth Mover Distance. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China, 3–7 November 2019; pp. 563–578.
52. Zhang, T.; Kishore, V.; Wu, F.; Weinberger, K.Q.; Artzi, Y. BERTScore: Evaluating Text Generation with BERT. In Proceedings of the International Conference on Learning Representations, Addis Ababa, Ethiopia, 26–30 April 2020; pp. 1–43.
53. Sellam, T.; Das, D.; Parikh, A. BLEURT: Learning Robust Metrics for Text Generation. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; pp. 7881–7892.
54. Doddington, G. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In Proceedings of the Second International Conference on Human Language Technology Research, San Francisco, CA, USA, 24 March 2002; pp. 138–145.
55. Lin, C.Y. Rouge: A package for automatic evaluation of summaries. In Proceedings of the Text Summarization Branches Out, Barcelona, Spain, 25 July 2004; pp. 74–81.
56. Vedantam, R.; Lawrence Zitnick, C.; Parikh, D. Cider: Consensus-based image description evaluation. In Proceedings of the IEEE conference on computer vision and pattern recognition, Boston, MA, USA, 7–12 June 2015; pp. 4566–4575.
57. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. 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), Minneapolis, MN, USA, 2–7 June 2019; pp. 4171–4186.
58. GAO, H.; WU, L.; ZHANG, H.; WEI, Z.; HU, P.; XU, F.; LONG, B. Triples-to-Text Generation with Reinforcement Learning Based Graph-augmented Structural Neural Networks. *arXiv* **2021**, arXiv:2111.10545.
59. Xie, T.; Wu, C.H.; Shi, P.; Zhong, R.; Scholak, T.; Yasunaga, M.; Wu, C.S.; Zhong, M.; Yin, P.; Wang, S.I.; et al. UnifiedSKG: Unifying and Multi-Tasking Structured Knowledge Grounding with Text-to-Text Language Models. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, Abu Dhabi, United Arab Emirates, 7–11 December 2022; pp. 602–631.
60. Ghazvininejad, M.; Karpukhin, V.; Gor, V.; Celikyilmaz, A. Discourse-Aware Soft Prompting for Text Generation. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, Abu Dhabi, United Arab Emirates, 7–11 December 2022; pp. 4570–4589.
61. Shen, S.; Fried, D.; Andreas, J.; Klein, D. Pragmatically Informative Text Generation. 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), Minneapolis, MN, USA, 2–7 June 2019; pp. 4060–4067.
62. An, S.; Li, Y.; Lin, Z.; Liu, Q.; Chen, B.; Fu, Q.; Chen, W.; Zheng, N.; Lou, J.G. Input-tuning: Adapting unfamiliar inputs to frozen pretrained models. *arXiv* **2022**, arXiv:2203.03131.
63. Hou, Z.; Salazar, J.; Polovets, G. Meta-learning the difference: Preparing large language models for efficient adaptation. *Trans. Assoc. Comput. Linguist.* **2022**, *10*, 1249–1265. [CrossRef]
64. Lin, Z.; Madotto, A.; Fung, P. Exploring Versatile Generative Language Model Via Parameter-Efficient Transfer Learning. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2020, Online, 16–20 November 2020; pp. 441–459.
65. Dušek, O.; Novikova, J.; Rieser, V. Findings of the E2E NLG Challenge. In Proceedings of the 11th International Conference on Natural Language Generation, Tilburg, The Netherlands, 5–8 November 2018; pp. 322–328.
66. Fleiss, J.L. Measuring nominal scale agreement among many raters. *Psychol. Bull.* **1971**, *76*, 378. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.