applied sciences

MDPI

Article

CTGGAN: Controllable Text Generation with Generative Adversarial Network

Zhe Yang, Yi Huang *, Yaqin Chen, Xiaoting Wu, Junlan Feng and Chao Deng

JIUTIAN Team, China Mobile Research, Beijing 100053, China; yangzhe@chinamobile.com (Z.Y.); chenyaqin@chinamobile.com (Y.C.); wuxiaoting@chinamobile.com (X.W.); fengjunlan@chinamobile.com (J.F.); dengchao@chinamobile.com (C.D.)

* Correspondence: huangyi@chinamobile.com

Abstract: Controllable Text Generation (CTG) aims to modify the output of a Language Model (LM) to meet specific constraints. For example, in a customer service conversation, responses from the agent should ideally be soothing and address the user's dissatisfaction or complaints. This imposes significant demands on controlling language model output. However, demerits exist among traditional methods. Promoting and fine-tuning language models exhibit the "hallucination" phenomenon and cannot guarantee complete adherence to constraints. Conditional language models (CLM), which map control codes into LM representations or latent space, require training the modified language models from scratch and a high amount of customized dataset is demanded. Decoding-time methods employ Bayesian Rules to modify the output of the LM or model constraints as a combination of energy functions and update the output along the low-energy direction. Both methods are confronted with the efficiency sampling problem. Moreover, there are no methods that consider the relation between constraints weights and the contexts, as is essential in actual applications such as customer service scenarios. To alleviate the problems mentioned above, we propose **C**ontrollable **T**ext **G**eneration with **G**enerative **A**dversarial **N**etworks (**CTGGAN**), which utilizes a language model with logits bias as the Generator to produce constrained text and employs the Discriminator with learnable constraint weight combinations to score and update the generation. We evaluate the method in the text completion task and Chinese customer service dialogues scenario, and our method shows superior performance in metrics such as PPL and Dist-3. In addition, CTGGAN also exhibits efficient decoding compared to other methods.

Keywords: controllable text generation; generative adversarial network; language model; GPT

check for updates

Citation: Yang, Z.; Huang, Y.; Chen, Y.; Wu, X.; Feng, J.; Deng, C. CTGGAN: Controllable Text Generation with Generative Adversarial Network. *Appl. Sci.* **2024**, *14*, 3106. https://doi.org/10.3390/app14073106

Academic Editor: Gianluigi Ferrari

Received: 25 January 2024
Revised: 26 February 2024
Accepted: 27 March 2024
Published: 8 April 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The purpose of controllable text generation is to integrate constraints into the original language models during the encoding process or the decoding process, so that the generated text can satisfy the pre-defined constraints [1–3]. For example, we aim to complete texts with positive sentiment, or describe contents that relate to the "large language models", etc., where "positive sentiment" and "related to..." are all constraints. Controllable text generation can be achieved undemandingly by a fine-tuning strategy, which utilizes "description of the constraint (e.g., 'express with a positive sentiment')" as the prompt, and consumes samples that satisfy the constraints as the training data to fine-tune the language model. However, this method can only modify the generated results to make them as close as possible to the constraints, but cannot ensure an absolute accuracy [4,5]. At the same time, during the fine-tuning process, it is susceptible to destroying the original output probability of the language model, resulting in catastrophic forgetting [6].

Conditional Language Model (CLM) is also a solution for CTG [7–14]. Generally, of the CLMs, "control codes" (e.g., types for topic or sentiment, or some preset special notations) that describe constraints are supplemented ahead of the source texts to predict a

conditional generation probability. Representations of "control codes" can be attached to texts both in a parallel and hierarchical manner. In addition, explicitly mapping constraints into latent space for control is another type of CLM. After mapping, a joint distribution for constraints and latent representation is built for CTG. Nevertheless, both types mentioned above require a complex neural network design and training from scratch where a specially tailored dataset is also demanded.

Decoding-time methods are training-free for CTG solutions. A common approach is to modify the original generation probability of the next token to the product with the probability from the constraint discriminator according to the Bayesian principle [15–18]. For example, the decoding probability of the next token "want" by the original language model is 0.5 and that of "like" is 0.4. If only the language model is used for decoding, then "want" should be selected as the next token. However, the discriminator's probability of identifying "want" is 0.2, while that of identifying "like" is 0.4. Therefore, under comprehensive consideration, when the constraints are met, the probability of selecting "want" for the next token is 0.1 (i.e., $0.5 \times 0.2 = 0.1$), while the probability of selecting "like" is 0.16, then "like" is finally selected as the decoding token. However, this approach requires to solve the normalized value of the probability. Considering the vocabulary size of the decoding space, a large number of samples must be used to obtain an approximate solution, making it time-consuming and labor-intensive. As a compromise method, if a small proportion of samples (i.e., the top N samples) are chosen for calculation, it will not fully guarantee that all outputs meet the constraints. Another family of methods [19–22] specifies an energy function by plugging in constraints, allowing for heterogeneous constraint to be combined with each other. Nevertheless, the sampling efficiency is still a challenge that needs to be addressed.

Moreover, none of the methods above explicitly model the relationship between the weights of constraints and the training samples (if the training process is necessary). Intuitively speaking, the constrained weights vary under different contexts which can be derived from training data, but all the methods attach static weights for the discriminators when the constraint classifiers are employed in the models.

The method we propose in this paper adopts the tunable bias which is similar to BOLT [22], but optimizes the bias using a GANs-based [23] approach instead where the generation with bias is measured and updated by Discriminator score. The Discriminator is composed of a learnable combination of constraints and provides a reward function to gauge the generation. After biases are tuned, the knowledge of the Discriminator will be integrated into the bias network (i.e., the Generator). In the end, when generating text, only the Generator part is needed to directly generate texts that meet attributes (e.g., sentiment control and fluency), and the decoding process is just similar to the original language model, which is very convenient. Our contributions include the following three points:

1. We propose to use GANs to combine the language model with various discriminators. Through the adversarial training process, the rewards derived from the discriminators can be integrated into the language model decoding process to correctly guide generations under the constraints.
2. We are the first to propose the hypothesis that the weight of various constraints should vary with the different contexts ahead, and design an algorithm to learn this dynamic weight through training samples.
3. We perform text completion tasks on 15 kinds of English prefixes, and also make evaluations on customer service dialogue scenarios with soothing words to verify our algorithm. Both of them demonstrate the effectiveness of the proposed method.

2. Related Work

There are two categories of algorithms to solve the CTG problem: conditional language models and decoding-time methods.

- **Conditional Language Models**: The conditional language models (CLMs) can be achieved with different implementations. For instance:

CTLR model [7] splices control codes with the text, and expects to learn the association between the generated text and the control codes through the language model. The control codes usually indicate a particular domain of the training sample, e.g., "Books", "Reviews", etc., or more complex, such as "Reviews Rating:5.0". The model used in CTLR is exactly a language model without any improvement.

GSUM [8] encodes the control codes (i.e., the guidance signals mentioned in the paper) and the texts separately with a shared transformer encoder. During decoding, both features of the control codes and the texts are used for attention calculations. Concretely, they are successively introduced into two consecutive cross-attention structures to complete the guided abstractive summarization.

GeDi [9] trains a conditional language model to guide another language model, e.g., GPT-2, to generate controlled texts. It uses a pair of opposite control codes, such as "positive" and "negative", to obtain the discriminative probabilities on both sentiments. When decoding, the Bayes rule is employed to get the true probabilities of tokens under the specific constraint.

Attribute alignment [10] holds the idea that adding control codes ahead of texts may break the originally learned sequential dependencies; therefore, it learns a projection function for both the key and value features of the control codes (which are derived from a language model) and fuses them into the key/value representations of the text in the corresponding layer for a modified attention calculation.

Contrastive Prefixes [11] uses a categorical distribution to map the text to attribute category Z, and further learns a projection network to represent it as $H(Z)$ as attribute prefix representation. For decoding, the prefix representation is attached ahead of the text feature so as to generate controlled results.

LantentOPs [12] and MacLaSa [13] employs samples with attribute labels to train a VAE model which maps the attributes to latent space Z. When generating, it first obtains the appropriate z through the ODE diffusion process, and finally uses the language model, i.e., the VAE decoder, to generate texts that satisfy the constraints.

Gu et al. [14] trains an auto-encoder framework where the encoder projects a sentence into latent space and the decoder, i.e., a fixed language model, reconstructs the same sentence with the latent values. For controllable generation, it establishes links between the latent space and a prior space (which obeys the Gaussian distribution) using normalizing flow and samples from the prior space to derive the final CTG results.

- **Decoding-time Methods**: Different from CLMs, decoding-time methods require no pre-trained language model training.

PPLM [15] uses an attribute model $p(a|x)$ which is learned from an autoregressive LM's top-level hidden layer to measure current context and modify the LM's gradient via gradient ascent with increasing the probability of the desired attribute at each step. In case of problems of repetitive and low-quality texts, the authors compute the mean over the Dist-N ($N \in \{1,2,3\}$) scores and discard samples with a mean score that is less than the preset threshold.

Based on Bayesian factorization, Fudge [16] multiplies the output probability of the original language model with that of the attribute discriminator (i.e., a constraint classifier) to re-rank the possibility of the next token. At each step, Fudge first samples 200 top tokens according to the language model's output, and for each token calculates its probability (the current context) with a discriminator. After that, the Bayesian rule mentioned above is employed to rewrite the probabilities of the next tokens (the top 200) and provides a new order to them.

DExperts [17] uses desired attributes (such as positive sentiment) and undesired attributes (such as negative sentiment) as a controller pair for CTG. By increasing the discrimination probability of desired attributes and reducing the probability of undesired attributes, the final generated results would satisfy the constraints. Concretely, the model calculates logits from both desired and undesired attributes and attaches the differences between them to the logits of the original language model. With the modification, a con-

straint generation with a high probability under the desired attribute and a low probability under the undesired will be derived.

In the paper [18], the authors use a contrastive generator to generate a positive sample that satisfies the constraints and several negative samples that do not meet the constraints, and use them to calculate the posterior probability of the constraints. By maximizing the product of language model generation probability and the above posterior probability, the algorithm makes the final generated result controllable. In order to increase the convergence of generated texts, an external discriminator is introduced to provide measurement of the current context.

Mireshghallah et al. [19] view the product of constraint experts (i.e., attribute discriminators, Hamming distance expert, BertScore expert, etc.) as a probabilistic energy model and then sample using a gradient-free Gibbs sampler to produce the sequences that minimize the energy function, with lower energy indicating more constraints being satisfied.

Kumar et al. [20] represent the energy functions as a Lagrangian and perform updates on a much smaller embedding space which allows longer sequence generation.

The iterative sampling of these methods is typically slow in practice. In order to improve sampling efficiency, Qin et al. [21] introduce a gradient-based MCMC method Langevin Dynamics and perform sampling by iteratively updating a continuous relaxation of text using gradients of the energy function to solve the problem of defining gradient with discrete sampling. Liu et al. [22] propose the BOLT method which adds a set of biases to the predicted logits of the pre-trained language model at each decoding step where the biases are tuned to steer the decoding process towards low-energy direction. The straight-through gradient estimator (STE) is employed to bypass the discrete token in the backward pass.

The above methods set a reasonable and feasible optimization direction for CTG, making the next step of sampling more directional, which can avoid the dilemma of solving the normalized probability value through a large number of samples required when modeling conditional language models. However, these methods modify the language model output during the decoding stage, making the decoding process relatively time-consuming. Especially in real-time application scenarios such as customer service conversations, the algorithms that rely on the correction of the decoding process are obviously unacceptable. At the same time, the methods above are based on the assumption that the weights of different constraints are fixed. Every time we generate text, we need to use empirical knowledge to set the proportions. It is obviously inconsistent with the real scenarios that the weights vary with the contexts ahead.

3. Materials and Methods

In this section, we first introduce two typical neural networks of language models (Section 3.1) and generative adversarial networks (Section 3.2) which are the bedrock of our work. After that, we introduce the details of the proposed method (Section 3.3) and explain how to improve the groundworks (Sections 3.1 and 3.2) for CTG adaption.

3.1. Introduction to Language Models

Represented by GPT-2 [24], a language model is typically constructed as a distribution over a text composed of n tokens: $t(t_1, t_2, ..., t_n)$, aiming to reflect the probability of the entire text as a whole.

For the text mentioned above, the probability calculation formula can be expressed as:

$$p(t(t_1, t_2, ..., t_n)) = p(t_1) * p(t_2|t_1) * ... * p(t_i|t_{1:i-1}) * ... * p(t_n|t_{1:n-1}), \quad (1)$$

In this formula, $p(t_i|t_{1:i-1})$ represents the probability of generating the i-th token, which is jointly determined by the previously generated tokens from 1 to $i-1$. In other words, the probability of the i-th token appearing in the sentence should be equal to the conditional probability given the occurrence of the previous tokens from 1 to $i-1$.

According to the **Maximum Likelihood Estimation** method, when the true sample is known as t_{true}, the goal of training a language model is to maximize the log-likelihood estimation, i.e., $log(p(t_{true}))$. When transformed into a loss function, the objective is to minimize:

$$loss = -log(p(t_{true})) \tag{2}$$

And the loss is typically computed using cross-entropy:

$$loss = CE(p(t_{true}), t_{true}) \tag{3}$$

Language model decoding is usually implemented token by token, where each decoding token is added to the end of the generated text, and then the next token is decoded. Greedy search or beam search is commonly employed for this process. Greedy search involves selecting the token with the maximum probability at each step, and thus decoding sequentially. Beam search is an improved algorithm that strikes a balance between greedy search and an exhaustive approach. It has a hyper-parameter called beam size, denoted as k. In the first decoding step, it selects the tokens corresponding to the top k probability values as the candidate tokens. In every subsequent step, it will decode k^2 possible tokens based on the previous k tokens (i.e., k current tokens for a previous token), and still choose k candidate tokens with top probability values. This process continues until the final token for the text reaches, and the combination of tokens with the maximum probability value is chosen as the generated text.

3.2. Introduction to GANs

Generative Adversarial Networks (GANs) [23] were initially introduced by Ian Goodfellow in 2014 for the image generation task. GANs comprise a Generator and a Discriminator. The Generator aims to produce samples that deceive the Discriminator into the status of being unable to distinguish between generated and real samples. In addition, the Discriminator tries to capture differences between generated and real samples. Through a zero-sum game (the adversarial training process), GANs reach a **Nash Equilibrium** where the optimal strategies for the Generator and Discriminator result in a balance, and the abilities for them are enhanced. As a sequence, the Discriminator makes a distinction between Generated and real data with a probability of $1/2$.

The GANs loss function is expressed as:

$$E_{x \sim x_{Gen}} log(1 - Dis(x)) + E_{x \sim x_{true}} log(Dis(x)). \tag{4}$$

Specifically, Generator loss is expressed as $min(E_{x \sim x_{Gen}} log(1 - Dis(x)))$ and Discriminator loss is expressed as $max(E_{x \sim x_{Gen}} log(1 - Dis(x)) + E_{x \sim x_{true}} log(Dis(x)))$.

3.3. Proposed Method

CTG is generally modeled as $p(Y|X, A)$, where X represents the preceding text (including prompts like "Please use soothing language to address the user's problem" or incomplete texts like "the potato..."). A refers to constraints, such as the requirement to reply with a soothing tone in the customer service scenario or attach a positive sentiment to sentences for the text completion task. Y represents the generated text under these constraints. In the case of a single constraint or a combination of fixed-weight constraints, the equation can be further expressed as :

$$p(Y|X, W_1A_1 + ...W_iA_i + ...W_nA_n), \tag{5}$$

where W_i represents the weight of constraint Ai in the overall constraint combination. These weights are fixed based on prior knowledge, satisfying the condition $\sum_{i=1}^{n}(W_i) = 1$.

In practical scenarios, constraint weights are often alterable and closely related to the context. For example, in a task like "complete the text with positive sentiment", the subsequent strength of positive sentiment depends on the preceding text. If the previous

text is "it is...", then the continuation tokens should have a strong positive sentiment, e.g., "happy". However, if the previous text is "it is great...", then a little strength of positive sentiment is sufficient, as "great" already provides a significant positive sentiment meaning. In customer service dialogues, the text constraints dynamically change as the conversation moves forward. For instance, if the user asks, "I topped up 50 yuan this morning, but I received a reminder about outstanding fees tonight, what's going on?", the staff's response should primarily focus on using a soothing reply style to address the user's dissatisfaction or concerns. If the user follows up with "Can you check my detailed expenses?", at this point, the staff only needs to perform a normal business query, as the user's negative emotions have almost been alleviated. The example illustrates that constraint weights in Equation (5) are continuously changing with the progress of the conversation and cannot be simply represented as a static value, i.e., W_i. Therefore, we propose a modification to the constraint conditions, where real-world CTG can be expressed as:

$$p(Y|X, \sum_{i=1}^{n} W(X, A_i) * A_i), \tag{6}$$

with $W(X, A_i)$ (a function of the context and current constraint) representing the current weight of constraint A_i.

For the neural network's design, as is shown in Figure 1, inspired by BOLT [22], we add logits bias to the original language model's outputs for correction. The corrected output is expressed as:

$$logits = logits_{LM} + logits_b, \tag{7}$$

where $logits_{LM}$ represents the original language model output, and $logits_b$ represents the correction to the language model. Training $logits_b$ directly with the corpus, similar to fine-tuning a language model, can alleviate the "hallucination" phenomenon but does not guarantee that generated text will always meet the constraints. Therefore, according to the modified CTG expression in Equation (6), we need to evaluate the logit results for constraint satisfaction.

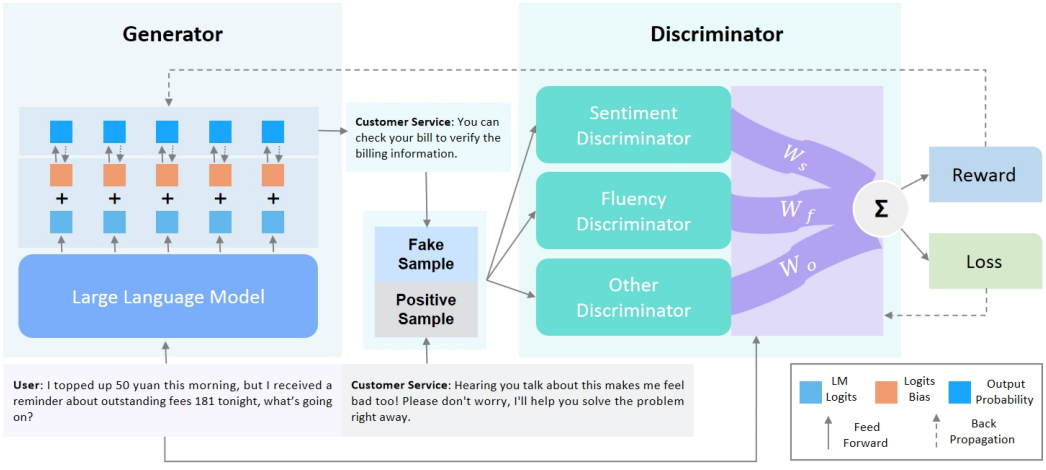

Figure 1. The overall framework of our model. We use the original LM encoder (such as GPT-2) to encode the text, leaving the original logits part unchanged (the dusty blue part in top-left of the picture), and adding the logits bias network (the orange part in top-left of the picture) as a modification to the original LM. The above network forms a complete Generator, which is used to generate text that satisfies the constraints. The right part is the Discriminator network, which is composed of multiple discriminators responsible for evaluating the text. The weight of each discriminator is represented by the text encoding and the discriminator embedding, determined by the similarity.

If continuing with the BOLT [22] method to satisfy constraints, which utilizes an energy function during inference to gradually adjust energy values to be as small as possible, we cannot learn $W(X, A_i)$ in Equation (6). Moreover, if using the CLM methods which model the true probability as $p = p_{LM} * p_{A_1}^{W(X,A_1)} * ... * p_{A_n}^{W(X,A_n)} / Z$, large-scale negative sampling will be required to approximate the normalization value Z. Therefore, we apply GANs to separate the $logits_b$ network from the constraint combination network $W(X, A_i) * A_i$. This not only avoids the issue of extensive negative sampling but also, after the model has been trained, allows expeditious CTG by combining the $logits_b$ network with the original output of the language model; namely, the efficiency of the decoding process would align perfectly with that of the original language model, without the need for additional operations such as negative sampling, making it more convenient.

Specifically, we use a language model, e.g., GPT-2, with biased $logits_b$ (modeled as Equation (7)) as the Generator part to generate text that satisfies the constraints combination. Additionally, the parameters of the original language model are frozen, and only those of the $logits_b$ network participate in back-propagation updates. The Discriminator part is designed based on the contrast learning approach where the positive samples come from the training corpus itself, while the negative samples are obtained by sampling text from the Generator network. We employ $W(X, A_i)$ for samples' scores calculation.

- **Generator and Optimization**: For the Generator part, i.e., $logits = logits_{LM} + logits_b$, the probability of output text can be expressed as:

$$\begin{aligned} p(Y|X, A) = softmax(logits_{LM_1} + logits_{b_1}) * ... \\ softmax(logits_{LM_i} + logits_{b_i}) * ... * softmax(logits_{LM_n} + logits_{b_n}). \end{aligned} \tag{8}$$

The goal of the Generator part is to make the distribution of generated text as close as possible to that of real corpora to deceive the Discriminator. In this context, greedy search is used to obtain the Generator's sampled result of current state, i.e.,

$$\begin{aligned} Gen(p_{sample}) = max(softmax(logits_{LM_1} + logits_{b_1})) * ... \\ max(softmax(logits_{LM_i} + logits_{b_i})) * ... * max(softmax(logits_{LM_n} + logits_{b_n})). \end{aligned} \tag{9}$$

The text generated by $argmax$ is denoted as Y_{gen}. As $argmax$ operator is not differentiable, the corresponding parameters cannot be updated according to the original GANs loss function. Therefore, we use the reinforcement learning approach (REINFORCE) [25] to redefine the loss function:

$$\nabla Gen = \frac{1}{n} \sum (logp(Y_{gen}|X, A)) * Reward(Y_{gen}). \tag{10}$$

Here, the reward function is calculated using $sigmoid(score_{gen} - score_{true})$, where $score_{gen}$ is the score of the generated sample Y_{gen}, and $score_{true}$ is the score of the real corpus data Y_{true}. The scores are determined through the Discriminator's constraint combination, i.e., $\sum W(X, A_i) * A_i$. Nevertheless, using this loss function alone may lead to instability in GANs training, failing to achieve the expected generation effect or even disrupting the distribution quality of the original language model. Therefore, without affecting the GANs loss function, we add the language model's inherent loss function (Equation (3)) as a supplement to the Generator loss function:

$$loss_{LM} = CE(p(Y|X, A), label) \tag{11}$$

where $p(Y|X, A)$ is the language model with logits bias (Equation (7)) from the previous text, and label is the real corpus tokens.

- **Discriminator and Optimization**: The Discriminator is responsible for scoring the qualities of texts, achieved through the combination of various constraint conditions. In this paper, for simplicity, two constraint conditions are set: positive sentiment

and fluency. The former can be implemented using a discriminative model such as BERT [26], where the input is a piece of text, and the output is the probability of being positive, i.e., $score_{pos} = BERT(Y)$. The latter can be calculated using perplexity (PPL), evaluating the fluency of the generated text with a language model, i.e., $score_{flu} = PPL(Y)$. The choice of the language model can be either the original language model (without logits bias in Equation (7)) of the Generator or another well-trained language model. In this paper, for convenience, we choose the original language model of the Generator for PPL calculation:

$$PPL(Y) = \sqrt[n]{\frac{1}{P(y_1) * P(y_2|y_1) * ... * P(y_n|y_1...y_{n-1})}} = e^{aver(CELoss)}, \quad (12)$$

where $aver(CELoss)$ means the average value of cross-entropy results which are obtained from predicted probabilities of the next tokens and their corresponding labels.

As mentioned earlier, the weights of constraints are different under context conditions, and these weights should be closely related to the context. Therefore, we model $W(X, A_i)$ as the similarity between the context and constraint A_i. Specifically, for the context X, we obtain its representation $Rep(X)$ through a language model and map it to the space of constraint Ai, i.e.:

$$Rep_X = Rep(X \rightarrow A_i) = W_i(LM(X)), \quad (13)$$

where W_i means a feed-forward network. The representation of constraint A_i is implemented through an embedding layer, i.e., $Emb(Ai)$. Consequently, the corresponding weight $W(X, A_i)$ for this constraint A_i can be expressed as:

$$W(X, A_i) = Sigmoid(W_i(LM(X)) * Emb(A_i)). \quad (14)$$

The scoring function can then be expressed as:

$$score(Y) = W(X, A_{sentiment}) * BERT(Y) + W(X, A_{fluency}) * PPL(Y). \quad (15)$$

Taking into account that the result of positive sentiment ($BERT(Y)$) is a probability value between 0 and 1, while fluency calculation results ($PPL(Y)$) are usually large, we attach a smaller coefficient, e.g., α, to the latter item. This ensures that positive sentiment and $PPL(Y)$ values are in a similar order of magnitude, preventing the possible neglect of positive sentiment's effect during training. Thus, the final scoring function is:

$$score(Y) = W(X, A_{sentiment}) * BERT(Y) + \alpha * W(X, A_{fluency}) * PPL(Y), \quad (16)$$

For the Discriminator's loss function, we draw inspiration from the contrast learning approach where a piece of text from the training corpus is considered as a positive sample, denoted as Y_{true}, and the text obtained by the Generator through greedy search (Equation (9)), i.e., Y_{gen}, is considered as a negative sample. Accordingly, the Discriminator's loss function can be expressed as:

$$loss(dis) = Sigmoid(score(Y_{true}) - score(Y_{gen})). \quad (17)$$

4. Experiments and Results

This section shows our experiments and results on two different types of datasets. The first one (Section 4.1) follows BOLT [22] with sentiment control for the text completion task where we select the same 15 prefixes to evaluate the positive sentiment and fluency scores of generated texts. The second one (Section 4.2) is from the Chinese customer service dialogue dataset which is set to respond to users with a soothing reply style so that can address their dissatisfaction and complaints.

4.1. Experiments on the Text-Completion Task

In the paper of PPLM [15], the authors propose 15 prefixes for the text completion task, where the generation style should correspond to the preset sentiment, such as positive, negative, or uncontrolled. Similar to BOLT [22], we follow those prefixes as the text prompts us to complete sentences for positive sentiment control. Different from baseline methods which design decoding algorithms for CTG and has no training necessary, the proposed method will need training samples to learn the parameters in Equations (8) and (14).

- **Training data**: We obtain 3000 samples from ChatGPT [27] as the positive sentiment controlled corpus, and select 1000 samples with high positive probabilities as the final dataset for the model's training, as is shown in Table 1. The prompt we use is, taking the prefix "The year is 1910" as an example, "Continue to write after 'The year is 1910' to express a positive sentiment". The generated corpus has a high quality to express positive sentiment to a certain extent; however, for the sake of positive probability measurement with quantity, we filter the low probability samples with a positive sentiment discriminator which is provided by the BOLT [22]. This operation is crucial as the only sentiment controller in our method setting is the positive sentiment discriminator, while the low positive probability samples will lead to a misgauge of the discriminator.

Table 1. Examples of training corpus generated with ChatGPT. High positive sentiment probability samples are kept for model training.

Prefix	ChatGPT Generation	Positive Probability
The year is 1910	The year is 1910, and a collective passion for learning and knowledge fuels a relentless pursuit of truth and enlightenment.	0.9767 (keep)
Once upon a time	Once upon a time, a genuine smile from a stranger brightened someone's day, spreading positivity far and wide.	0.9961 (keep)
The country	The country had a strong emphasis on education and intellectual growth, fostering a culture of learning and knowledge.	0.9466 (keep)
The movie	The movie we attended the premiere of left us star-struck, walking the red carpet and witnessing the magic of cinema unfold before our eyes.	0.2231 (deleted)

- **Model's setting**: Similar to BOLT [22], we utilize GPT2-large (https://huggingface.co/gpt2-large, accessed on 5 December 2023) as the basis of the Generator part and also the fluency measurement model where the latter is calculated by Perplexity (PPL). As for positive sentiment judgment, we employ a BERT classifier model which is trained after "yelp polarity" dataset (https://www.yelp.com/dataset, accessed on 5 December 2023). During GANs training, parameters of both the models above (i.e., GPT2-large and BERT classifier) are frozen with no parameters updated. We only train parameters like bias network (i.e., $logtis_b$ in Equation (8)), embedding layers, and other linked parts (i.e., $W(X, A)$ in Equation (14)).
- **Evaluation metrics**: As the Table 2 shows, we evaluate the generated texts from four aspects: internal positive classifier evaluation (Int.Clsf) measures the positive sentiment score as the form of the positive probability derived from the BERT classifier ($BERT(Y)$ mentioned in Equation (15)); external positive classifier evaluation (Ext.Clsf) also calculates the positive probability but using an external classifier obtained from huggingface (https://huggingface.co/VictorSanh/roberta-base-finetuned-yelp-polarity, accessed on 5 December 2023); perplexity (PPL) evaluates the fluency of the generated text which is calculated by the GPT2-XL (https://huggingface.co/gpt2-xl, accessed on 5 December 2023) model according to Equation (12); the average occurrences of distinct

trigrams (Dist-3) measures the degree of diversity of the generation which is reckoned up by NLTK (https://www.nltk.org/, accessed on 5 December 2023) package.

- **Results**: As can be seen from the Table 2, our method surpasses the traditional methods (each of them obtains the generated results from the corresponding decoding strategy straightforwardly on the 15 prefixes mentioned above) in the two metrics of PPL and $Dist-3$. Specifically, The PPL value is better than other methods by an average of 14.58, and $Dist-3$ is better than other methods by an average of 0.35. In addition, the probability value of the external classifier (Ext.Clsf) can also reach a comparable level to traditional methods. Our method is weaker than methods such as "MuCola | Mix&Match | BOLT" in terms of internal classifier probability (Int.Clsf). It is because our method considers not only positive sentiment, but also the fluency of the sentence, so it needs to learn the weights of those two stuffs from training positive samples which is different from the traditional methods that only pursue high positive sentiment probability. Moreover, there exist gaps between the training samples we employ and the so-called "perfect" samples (whose positive probabilities can reach 1); thus, from the training perspective, the positive score ceiling of our method will be lower than that of other baseline methods.

Compared with those energy-based methods listed in Table 2, our method also has a significant advantage of small time consumption in the decoding stage. To be detailed, our method is based on training, and the time is mainly spent in the training phase. By designing a learnable network, training samples are consumed in the process of optimizing, so that the network can learn reasonable parameters that meet the constraint conditions. In this way, when decoding, only the Generator part of the network is needed, and the text that satisfies the constraints can be decoded as easily as the original language model, without the need to constantly modify the output of the language model along the direction of minimum energy value.

Table 2. Results on positive sentiment control. Int.Clsf means the positive sentiment score measured by $BERT(Y)$ in Equation (15), higher (↑) is better. Ext.Clsf is the positive sentiment score measured by an external sentiment classifier, higher is better. PPL measures the fluency of the generated texts calculated by Equation (12), lower (↓) is better. Dist-3 evaluates the average occurrences of distinct trigrams, higher is better.

Model	Int.Clsf (↑)	Ext.Clsf (↑)	PPL (↓)	Dist-3 (↑)
COLD [21]	61.46	55.10	9.09	0.30
MuCola [20]	93.22	86.55	11.36	0.55
Mix&Match [19]	96.09	84.98	66.75	0.82
BOLT [22]	95.78	80.12	8.12	0.65
Our Model	81.56	80.73	9.25	0.93

- **Ablation study**: We only use the Generator part to train the above positive samples through the loss function of the language model (i.e., Equation (11)). The results are shown in Figure 2. It can be seen that without Discriminator to provide rewards to optimize the Generator network and guarantee the generated results, the evaluation metrics will decline to varying degrees, to be detailed, there are 4.6%, 6.8%, and 8.5% deterioration on metrics of Int.Clsf, Ext.Clsf, and Dist-3, and PPL value increases by 0.61 which means a worse fluency. Hence, the Discriminator plays an important role in guiding the Generator to produce high-quality constrained texts. In addition, it validates that CTG cannot be ensured by using only language model fine-tuning.

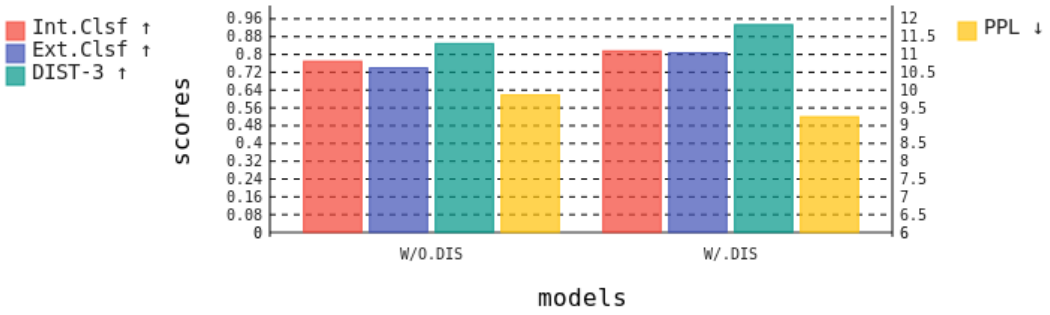

Figure 2. Ablation results in 15 prefixes evaluation. W/O.DIS means generating texts with only Generator training.

4.2. Experiments on Chinese Customer Service Dialogue

Chinese customer service dialogue data comes from materials collected by online companies. In the daily operation process, users will encounter various usage problems, such as broadband failures, fee doubts, package inquiries, etc., which will result in a certain degree of dissatisfaction or complaints. In this case, customer service executives need to use soothing words to first calm the users' dissatisfaction, and then provide reasonable explanations or propose feasible solutions to the users' problems. A customer service that satisfies users can always calm users' negative emotions in the first place; provide positive guidance to users' psychology; patiently analyze and inform the causes of failures step by step; proactively propose certain compensatory measures for users, and repair the fault problems at the end. The above-mentioned soothing words are often industry-specific responses that customer service staff have accumulated through long-term work experiences and detailed analysis of users' emotions. Soothing words often contain strong honorifics, which can send a signal of goodwill to users and give them an attitude of "eager to solve the problem" in the first place. At the same time, soothing words have obvious "empathy" skills. For example, when a user encounters a problem, the customer service staff will say "I'm sorry for your experience. No matter who encounters such a problem, they will be very angry. We express our deep understanding and sorry". Such words can often overcome the estrangement with users, and gain their understanding.

We have collected 500 complete customer service dialogue materials with soothing words. Each dialogue consists of several rounds of conversations in the format of "User: [User Question]; Customer Service: [Customer Service Reply]". User problems often involve some complaints, and customer service replies can usually solve these problems better. Soothing words are often different from the common positive emotional responses, and cannot be judged using a simple discriminative model trained on the positive and negative emotional dataset, for example, the English Yelp dataset, the Chinese Weibo emotional dataset (https://smp2020ewect.github.io/, accessed on 5 December 2023), etc. , thus we first need to customize a discriminative model that satisfies the judgment of soothing words.

- **Soothing reply discriminator**: We harness the Chinese BERT model (https://github.com/ymcui/Chinese-BERT-wwm, accessed on 5 December 2023) as the basic model for the soothing word discriminator. The positive samples, a total of 5000 sentences with higher quality, are selected from the customer service staff's responses in the customer service dialogue materials. We modify some of the sentences, for example, replacing the Chinese word "你 (you)" (non-honorable word) with "您 (you)" (honorable word). In order to get negative samples, that is, replies without soothing words, we use the ChatGLM-3 [28] large language model to rewrite the positive samples. The rewriting prompt we use is "将下列文本换成简短精炼的说法，意思保持不变，不要出现谦辞敬辞。(Change the following text into a short and concise statement, the meaning remains the same, and do not appear to be polite or respectful.)". Table 3 displays some case comparisons before and after ChatGLM-3's modifications which indicate that after modifications with the ChatGLM-3 model, the corresponding ex-

pressions are more frigid in terms of the emotion and do not have a soothing style. If a similar reply manner is used, it will aggravate the user's dissatisfaction.

Table 3. Examples of negative samples (none-soothing-expression) generated with ChatGLM-3 model.

Scenario	Soothing-Expression	ChatGLM-3 Gennerated
Change the mobile data	直接给您改不了，因为您套餐一百零八送的带宽是多少兆，它就是多少兆的. (I am sorry that I can't change it for you directly, because it is equal to the number of megabytes of bandwidth which is given by your package of one hundred and eight.)	直接改不了，因为套餐一百零八送的带宽就是多少兆. (I can't change it directly, because the bandwidth of the package of one hundred and eight is so many megabytes.)
Reset the password	那没有关系，我把重置密码的方法给您发过来，您可以发送短信重置密码. (That's okay. I will send you the method to reset your password, and you can send a text message to reset it.)	没问题，我将重置密码的方法告诉你，你发短信即可重新设置密码. (No problem. I'll show you how to reset your password, and you can reset it by texting it.)
Cancel the account	销户的话，您需要到营业厅办理，首先要看看这个号码上有没有活动，如果没有活动，现场销户，如果有活动，您需要和营业厅协商中断违约金，才可以中断. (If you want to cancel your account, you need to go to the business office to handle it. First, it is advisable for you to check if there is any activity on this number. If there is no activity, you can cancel the account on the spot. If there is activity, you need to negotiate with the business office for liquidated damages before you can cancel.)	如果你要销户，你需要到营业厅办理。首先查看号码是否有活动，无活动则现场销户，有活动则需与营业厅协商中断违约金. (Cancelling the account, you need to go to the business hall to do it. First check whether there is any activity on the number. If there is no activity, the account will be canceled on the spot. If there is activity, you need to negotiate with the business office for a penalty for interruption.)

- **Customer service dialogue training**: We utilize the Chinese GPT-2 (https://github.com/Morizeyao/GPT2-Chinese, accessed on 5 December 2023) model as the basic part of the Generator. Similarly, this model is also used as the fluency discriminator, that is, to calculate the PPL value of the generated text. During the training process of the Generator, it should be noted that tokens irrelevant to the customer service replies will be marked with a special **MASK** notation. In this way, when calculating the language model loss in Equation (11), the cross-entropy function of these tokens will be ignored and only tokens associated with customer service reply contents will be calculated.
- **Results**: Table 4 lists several cases generated under constraints, where W/. DIS refers to the results generated using our method, and W/O.DIS refers to those derived by training only the Generator (similar to the ablation study in Section 4.1). It can be seen that our method can generate replies that satisfy the custom service scenario by and large. Specifically, the response logic is basically correct (for example, the reply of "canceled your subscription" corresponds to the context of "I have canceled this package" in case 5), and it is generally guaranteed to generate replies with soothing words (for example, "Thank you" in case 4, and "anything I can do for you" in case 5). Nevertheless, the text generated by W/O. DIS will obviously retell the words from context ahead, e.g., "spend one yuan to get a 50-min call package" in case 3. Therefore, when using a lightweight language model like GPT-2 for dialogue training, our method can generate relatively controllable responses.

Table 4. Examples of generated texts in Chinese customer service scenario.

Prompt	W/O.DIS	W/.DIS
用户:喂你好，帮我查下我流量呢? (hello, please check the mobile data for me.)	号码是您的吗？(Is the phone number yours?)	号码已经给您办理了，请稍后联系我们查询。(The number has been processed for you, you can contact us later.)
用户:我宽带用不了了，电视也不能看。(I can't use my broadband, and I can't watch TV)	号码是您的吗？(Is the phone number yours?)	局部网络故障，无法使用（Localized network failure and unavailability）
用户：你好，我的通话用超了，是不是可以花一块钱办一个五十分钟的通话包？(I have overused my calls, can I get a 50-min call package with one yuan?)	先生，您的通话包是五十分钟，是不是可以花一块钱办一个五十分钟的通话包. (Sir, your call package is 50 min, is it possible to spend one yuan to get a 50-min call package?)	先生，请问还有其他可以帮助您的吗？您可以通过以下方式进行咨询 (Sir, is there anything else that can help you? You can make an inquiry in the following ways)
用户：喂我问一下我现在这个WIFI宽带是没用了哇？(Hey, I'm here to ask if the WIFI broadband is useless.)	您的wifi是有用的吗？(Is your wifi still in use?)	话务员给您开通一下WIFI，然后您点开一下WIFI就可以使用了。感谢您。(The operator will turn on the WIFI for you, and then you can use it by clicking on the WIFI. Thank you.)
用户：把我这个套餐都取消了，怎么又变成九十多了呢？(I have canceled this package, why did it become more than 90 again?) 客服：这个是扣月租，但是我们这边显示您并没用更改过套餐，您这个是六月十二号办的，这个八十九块钱套餐，七月一号生效的。(This is a deduction of monthly rent, but it shows that you haven't changed your plan. You applied 89 yuan package on June 12th, and it took effect on July 1st.)用户：你给我取消。(Cancel the plan for me.)	来电，您这个套餐是六月十二号办的。(Caller ID shows your package is applied on June 12th.)	来电显示您已经取消订购了，请问还有其他需要帮助的吗？(Caller ID shows that you have canceled your subscription. Is there anything I can do for you?)

5. Discussion and Limitations

The controllable text generation algorithm we propose employs ideas of GANs to separate the constrained text generation process from that of constraints measurement. The Generator, which is a parameter-frozen language model (i.e., GPT-2) with a logits bias, uses positive samples (i.e., the training data) and rewards from the Discriminator (i.e., scores on both positive samples and those generated by the Generator) to optimize the bias network. In addition, the Discriminator provides learnable weights for different discriminators (i.e., the sentiment and the fluency discriminators in the paper) and a reward function to gauge the difference between positive/negative pairs. The design of the learnable weights comprises of similarities (i.e., the sigmoid function after the feedforward network) between context representation and discriminators' embedding, and affords a dynamic constraints combination strategy; hence, the generated texts will be various and consistent with the contexts ahead. Therefore, when controlling text generation under a specific context, our algorithm will take into account the different degrees of contribution for each Discriminator, which makes the control of text generation more refined, making up for the shortcomings of the single constraint condition or fixed weight combination constraint condition of the

traditional algorithm, so the text generated by the proposed method matches the higher degree of its context ahead. Moreover, during the decoding stage, only the Generator part is necessary for CTG which brings forth the efficiency for text generation.

We conduct experiments on the text completion task and evaluate several decoding-time baselines. Concretely, we make prompts for ChatGPT to generate and filter 1000 samples with high positive sentiment probabilities for CTGGAN's training. After that, we employ the Generator to complete the 15 prefixes and evaluate the results on four metrics (i.e., Int.Clsf, Ext.Clsf, PPL, and Dist-3). Compared with the baselines, our method shows superiority on PPL and Dist-3 (i.e., 14.58 and 35% improvements), and displays analogous performance on other metrics. We make an ablation study as well which discards the Discriminator and only trains the Generator through the typical loss function of the language model. The result indicates that without judgments from the discriminators, all the metrics will degrade to a certain extent (i.e., 4.6%, 6.8%, 0.61, and 8.5% deterioration on the corresponding metrics). In addition, we evaluate the method in the Chinese customer service dialogue scenario, and make the conclusion that without the Discriminator, the generated responses will have repetitions more often.

Unfortunately, our algorithm needs to secure negative samples through a greedy search on the Generator's outputs, which means an entire inference process of the language model must be implemented at every training step. It inevitably leads to the problem of low training efficiency especially when the parameters' size of the Generator gets larger. Therefore, in future work, we will explore better sampling strategies to make model training more efficient.

6. Conclusions

We propose **CTGGAN**, a controllable text generation algorithm that combines the advantage of language model decoding with various discriminators in a learnable and skillful way. Specifically, CTGGAN utilizes the language model with a bias network as the Generator to generate text that meets the constraints, and compounds the sentiment discriminator and fluency discriminator as the Discriminator part to score the quality of the generated text and guide the optimization of the Generator part. We use the adversarial learning idea of GANs to iteratively learn and optimize the entire model. After the model's training, it only needs the Generator part to generate texts that meet constraints, which does not require continuously correcting the text during decoding along the direction of maximizing the constraint possibilities and is as efficient as the original language model. We selected 15 English prefixes for the text completion task and compared them with the baseline methods, which reflects the superiority of our algorithm. At the same time, we conduct algorithm evaluation on the Chinese customer service conversation dataset. The results also show that our algorithm has certain feasibility in soothing words scenarios of customer service dialogues.

Author Contributions: Methodology, Z.Y., Y.H., Y.C., and X.W.; Validation, Y.H., J.F., and C.D.; Formal analysis, Z.Y. and Y.C.; Investigation, Z.Y. and Y.C.; Data curation, Z.Y. and Y.C.; Writing—original draft preparation, Z.Y., Y.C., and X.W.; Writing—review and editing, Y.H.; Visualization, Z.Y. and X.W.; Supervision, J.F. and C.D.; Project administration, Y.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by China Mobile Holistic Artificial Intelligence Major Project Funding (R22105ZS, R22105ZSC01), the National Key R&D Program of China (2021ZD0140408), and the Beijing Natural Science Foundation (L222006).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Training data are derived from ChatGPT or ChatGLM-3 generation which can be replicable according to our prompts mentioned in Section 4, further inquiries can be directed to the corresponding author.

Conflicts of Interest: Authors Zhe Yang, Yi Huang, Yaqin Chen, Xiaoting Wu, Junlan Feng and Chao Deng were employees of China Mobile Research. The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Prabhumoye, S.; Black, A.W.; Salakhutdinov, R. Exploring Controllable Text Generation Techniques. In Proceedings of the 28th International Conference on Computational Linguistics, Barcelona, Spain, 8–13 December 2020; pp. 1–14. [CrossRef]
2. Zhang, H.; Song, H.; Li, S.; Zhou, M.; Song, D. A Survey of Controllable Text Generation Using Transformer-based Pre-trained Language Models. *ACM Comput. Surv.* **2022**, *56*, 64. [CrossRef]
3. Zhu, L.; Xu, Y.; Zhu, Z.; Bao, Y.; Kong, X. Fine-Grained Sentiment-Controlled Text Generation Approach Based on Pre-Trained Language Model. *Appl. Sci.* **2023**, *13*, 264. [CrossRef]
4. Gururangan, S.; Marasović, A.; Swayamdipta, S.; Lo, K.; Beltagy, I.; Downey, D.; Smith, N.A. Don't Stop Pretraining: Adapt Language Models to Domains and Tasks. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020 ; pp. 8342–8360. [CrossRef]
5. Huang, Y.; Wu, X.; Hu, W.; Feng, J.; Deng, C. State-Aware Adversarial Training for Utterance-Level Dialogue Generation. In Proceedings of the Towards Semi-Supervised and Reinforced Task-Oriented Dialog Systems (SereTOD), Abu Dhabi, UAE, Beijing, China (Hybrid), 7 December 2022; pp. 62–74. [CrossRef]
6. Gan, Y.; Lu, G.; Su, Z.; Wang, L.; Zhou, J.; Jiang, J.; Chen, D. A Joint Domain-Specific Pre-Training Method Based on Data Enhancement. *Appl. Sci.* **2023**, *13*, 4115. [CrossRef]
7. Keskar, N.S.; McCann, B.; Varshney, L.R.; Xiong, C.; Socher, R. CTRL: A Conditional Transformer Language Model for Controllable Generation. *arXiv* **2019**, arXiv:1909.05858.
8. Dou, Z.Y.; Liu, P.; Hayashi, H.; Jiang, Z.; Neubig, G. GSum: A General Framework for Guided Neural Abstractive Summarization. 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. 4830–4842. [CrossRef]
9. Krause, B.; Gotmare, A.D.; McCann, B.; Keskar, N.S.; Joty, S.; Socher, R.; Rajani, N.F. GeDi: Generative Discriminator Guided Sequence Generation. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2021, Punta Cana, Dominican Republic, 7–11 November 2021; pp. 4929–4952. [CrossRef]
10. Yu, D.; Yu, Z.; Sagae, K. Attribute Alignment: Controlling Text Generation from Pre-trained Language Models. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2021, Punta Cana, Dominican Republic, 7–11 November 2021; pp. 2251–2268. [CrossRef]
11. Qian, J.; Dong, L.; Shen, Y.; Wei, F.; Chen, W. Controllable Natural Language Generation with Contrastive Prefixes. In Proceedings of the Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, 22–27 May 2022; pp. 2912–2924. [CrossRef]
12. Liu, G.; Feng, Z.; Gao, Y.; Yang, Z.; Liang, X.; Bao, J.; He, X.; Cui, S.; Li, Z.; Hu, Z. Composable Text Controls in Latent Space with ODEs. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Singapore, 6–10 December 2023; pp. 16543–16570. [CrossRef]
13. Ding, H.; Pang, L.; Wei, Z.; Shen, H.; Cheng, X.; Chua, T.S. MacLaSa: Multi-Aspect Controllable Text Generation via Efficient Sampling from Compact Latent Space. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, 6–10 December 2023; pp. 4424–4436. [CrossRef]
14. Gu, Y.; Feng, X.; Ma, S.; Zhang, L.; Gong, H.; Zhong, W.; Qin, B. Controllable Text Generation via Probability Density Estimation in the Latent Space. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Toronto, ON, Canada, 9–14 July 2023; pp. 12590–12616. [CrossRef]
15. Dathathri, S.; Madotto, A.; Lan, J.; Hung, J.; Frank, E.; Molino, P.; Yosinski, J.; Liu, R. Plug and Play Language Models: A Simple Approach to Controlled Text Generation. In Proceedings of the International Conference on Learning Representations, Addis Ababa, Ethiopia, 30 April 2020.
16. Yang, K.; Klein, D. FUDGE: Controlled Text Generation With Future Discriminators. 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. 3511–3535. [CrossRef]
17. Liu, A.; Sap, M.; Lu, X.; Swayamdipta, S.; Bhagavatula, C.; Smith, N.A.; Choi, Y. DExperts: Decoding-Time Controlled Text Generation with Experts and Anti-Experts. 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. 6691–6706. [CrossRef]
18. Liu, G.; Li, Y.; Guo, Y.; Luo, X.; Wang, B. Multi-Attribute Controlled Text Generation with Contrastive-Generator and External-Discriminator. In Proceedings of the 29th International Conference on Computational Linguistics, Gyeongju, Republic of Korea, 12–17 October 2022; pp. 5904–5913.
19. Mireshghallah, F.; Goyal, K.; Berg-Kirkpatrick, T. Mix and Match: Learning-free Controllable Text Generationusing Energy Language Models. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Dublin, Ireland, 22–27 May 2022; pp. 401–415.

20. Kumar, S.; Paria, B.; Tsvetkov, Y. Gradient-based Constrained Sampling from 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. 2251–2277. [CrossRef]
21. Qin, L.; Welleck, S.; Khashabi, D.; Choi, Y. COLD Decoding: Energy-based Constrained Text Generation with Langevin Dynamics. In *Advances in Neural Information Processing Systems*; Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2022; Volume 35, pp. 9538–9551.
22. Liu, X.; Khalifa, M.; Wang, L. BOLT: Fast Energy-based Controlled Text Generation with Tunable Biases. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Toronto, ON, Canada, 9–14 July 2023; pp. 186–200. [CrossRef]
23. Goodfellow, I.J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.C.; Bengio, Y. Generative Adversarial Nets. In Proceedings of the Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014.
24. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language Models are Unsupervised Multitask Learners. *OpenAI Blog* **2019**, *1*, 9.
25. Wang, J.; Yu, L.; Zhang, W.; Gong, Y.; Xu, Y.; Wang, B.; Zhang, P.; Zhang, D. IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Tokyo, Japan, 7–11 August 2017.
26. 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. [CrossRef]
27. OpenAI; Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F.L.; Almeida, D.; Altenschmidt, J.; Altman, S.; et al. GPT-4 Technical Report. *arXiv* **2023**, arXiv:2303.08774.
28. Du, Z.; Qian, Y.; Liu, X.; Ding, M.; Qiu, J.; Yang, Z.; Tang, J. GLM: General Language Model Pretraining with Autoregressive Blank Infilling. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Dublin, Ireland, 22–27 May 2022; pp. 320–335.

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.