Friendly Conditional Text Generator

Noriaki Kawamae
NTT Comware
kawamae@gmail.com

ABSTRACT

Our goal is to control text generation with more fine-grained conditions at lower computational cost than is possible with current alternatives; these conditions are attributes (i.e., multiple codes and free-text). As large-scale pre-trained language models (PLMs) offer excellent performance in free-form text generation, we explore efficient architectures and training schemes that can best leverage PLMs. Our framework, Friendly Conditional Text Generator (FCTG), introduces a multi-view attention (MVA) mechanism and two training tasks, Masked Attribute Modeling (MAM) and Attribute Linguistic Matching (ALM), to direct various PLMs via modalities between the text and its attributes. The motivation of FCTG is to map texts and attributes into a shared space, and bridge their modality gaps, as the texts and attributes reside in different regions of semantic space. To avoid catastrophic forgetting, learn modality-free embedded representations, and direct PLMs in this space, FCTG applies MAM to learn attribute representations, maps them in the same space as text through MVA, and optimizes their alignment in this space via ALM. Experiments on publicly available datasets show that FCTG outperforms baselines over higher level conditions at lower computation cost.

CCS CONCEPTS

• **Computing methodologies → Natural language generation**.

KEYWORDS

Conditional Text Generation, Neural Language Model, Neural Language Generation, Transformer

ACM Reference Format:
Noriaki Kawamae. 2023. Friendly Conditional Text Generator. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining (WSDM '23), February 27–March 3, 2023, Singapore, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3539597.3570364

1 INTRODUCTION

Transformer-based neural language models (NLMs) [11, 22, 31, 39, 46, 51] have demonstrated outstanding performance in a variety of natural language processing tasks. Among these tasks, text generation enables computers to create fluent expressions. This task's importance has been recognized by both industry and academic communities as promising [9, 17], who have applied NLMs and demonstrated their natural language generation (NLG) capabilities. Unfortunately, these pre-trained NLMs (PLMs) are difficult to control beyond providing prompts for the continuation of the generation process [6, 7, 28]. At first glance, large-scale PLMs appear to be a natural fit for NLG since their pre-training objectives are often derived from language modeling [15].

PLMs generate tokens by simply iteratively sampling the next token, but many applications demand control of the output to satisfy setting attributes such as the topic, category, and/or author. For example, consider PLMs being used to generate AI articles for researchers or the public; the latter require explanations of technical terms that are well known to the researchers and so plain terms should be used as much as possible. Although PLMs have achieved excellent performance in free-form text generation with promising capabilities, they cannot be directly employed where text generation must satisfy specific lexical constraints [54]. Moreover, users cannot easily control particular aspects of the generated text [17]. Once such models are trained, controlling the attributes of the generated text becomes difficult without modifying the model architecture to allow for the input of extra attributes or fine-tuning with attribute-specific data [9, 17, 55], both of which entail the significant cost of retraining. If we can introduce complex combinations of various codes or free-text guide to text generation, significantly more services and applications will become possible.

Motivated by the above background, we propose **F**riendly **C**onditional **T**ext Generator (FCTG), a simple framework that guides PLMs to generate text following given conditions. These conditions are attributes including both free-text and more fine-grained codes (e.g., product name/category/brand, customer id or various keywords) than basic codes (e.g., topic field and sentiment). Given the possibility of 1) interpreting relationships between text and its attributes as modalities, and 2) bridging the modality gaps to steer PLMs towards the target domains, we explore how to learn modality-free representations and apply them in support of controllable generation tasks. These motivations yield our framework, Friendly Conditional Text Generator (FCTG), a multi-view attention (MVA) mechanism and the training tasks of Masked Attribute Modeling (MAM) and Attribute Linguistic Matching (ALM). MVA enables FCTG to offer bidirectional attention flow between attributes and words and project them into the same space which permits the direct evaluation of their similarity, while steering PLMs without the problem of catastrophic forgetting [33, 34]; MAM learns representations of attributes, while ALM aligns text-level attributes in the space.

Experiments confirm the advances secured by FCTG's simple framework and lower computation cost.

Theoretical contribution: MVA, MAM and ALM allow FCTG to capture all the interactions among words and attributes, learn modality-free representations, and steer PLMs to generate text in compliance with given conditions as shown in 6.5 and 6.6.

Practical contributions: FCTG can 1) control text generation with various detailed conditions as shown in 6.4, 2) accommodate vocabulary growth so as to support new datasets as shown in 6.4, and 3) leverage PLMs using few parameters and at low computation cost by using attribute-tuning, as shown in 6.7.

2 PREVIOUS WORK

Recently, pre-trained Transformer [46] based language models [11, 22, 31, 39, 51], PLMs, have yielded great advances in NLP tasks. They are pre-trained on large-scale unlabeled text corpora, and then fine-tuned to suit the downstream task. They use a multi-layer attention mechanism [4] to allow fully-connected self-attention to the full context in a computationally efficient manner. For example, they have higher and deeper structures (up to 48 layers in GPT-2) and are more effective in leveraging large-scale datasets (more than 100 million training instances) than RNN [12] or LSTM [16]-based approaches. This shared high-level idea has been explored in the literature for different unsupervised pre-training objectives. Given an input token sequence, UNIfied pre-trained Language Model (UniLM) [13] is directed towards natural language understanding and NLG tasks as it employs a shared Transformer network and utilizes specialized self-attention masks to control what context the prediction is conditioned on.

While large-scale Transformer-based models [11, 24, 46, 51] have shown promising text generation capabilities, they cannot be directly employed to generate text under specified lexical constraints [54]. This architecture does not yet provide sufficient advantage in text generation [44, 45], and it is difficult for users to control generation with a view to particular aspects [17]. Although many of those models are pre-trained on large-scale corpora, they are limited in specific domains as they either utilize domain-specific priors or were not designed to generate texts in different domains or styles. To address this challenge, current controlled text generation methods either fine tune existing models with Reinforcement Learning [55], training Generative Adversarial Networks [52], or train conditional generative models [17]. Conditional Transformer Language Model for Controllable Generation (CTRL) [17] generates text conditioned on 50 different control codes. However, these codes must be predefined which limits flexibility, and are only used once at the beginning to condition the generation of the rest of the document. To address these shortcomings, MEGATRON-CNTRL [50] adds control to text generation by incorporating an external knowledge base. POINTER [54] operates by progressively inserting new tokens between existing tokens in a parallel manner. Approaches based on insertion [43], GAN [10, 47, 52, 53] and Reinforcement learning [55] have been applied to the text generation task. Pre-train and Plug-in Variational Auto-Encoder [14] employs VAE [19], and decouples the text generation module from the condition representation module. The Plug and Play Language Model (PPLM) [9] introduces plugging as a discriminator [36] that combines a PLM with attribute classifiers that guide text generation.

As text generation tasks involve an enormous output space, it is often too expensive to curate clean quality data in sufficient quantity and estimate the conditional distribution correctly over many attributes. OAG-BERT [30] leverages document attributes/metadata in a pre-training step. Blending Generative Model (BGM) [28] enhances GeDi [20] with additional capabilities to support multi-topic generation with continuous weighting. Although they are expressive, they are less effective than previous alternatives in controlling generation at specific points. This is due to the prompt's influence being negatively correlated with the distance from the prompt to the next predicted token [56], which makes prompting for non-adjacent text difficult. Content-Conditioner (CoCon) [7] incorporates the representations of input content into encoded text representations.

These studies motivate us to explore simple but effective frameworks for conditional text generation. Like other NLMs, our task is formulated as an autoregressive (AR) language model using the multi-layer Transformer. To allow FCTG to focus on the modality between attributes and words, a unique capability, we derive MAM, MVA and ALM; to realize conditioned text generation, and fine-tune PLMs by adding steerable layers like prompt learning. Unlike UniLM, MVA allows attributes to attend to linguistic tokens. While Li and Liang [26] propose prefix-tuning (Prefix) as an alternative to full fine-tuning for conditional generation tasks, only prefix parameters are trainable, FCTG observes both attributes and words, but trains them using the fewest parameters possible. It differs from CoCon [7] and NRP [6] in that it controls a greater variety of attention flows automatically, uses continuous and contextualized representations in training, and updates both attributes and words.

3 PROBLEM FORMULATION

Controllable generation entails modeling that can be formulated using the conditional distribution $P(\mathbf{x}|c)$, concatenating conditions like the prefix[6, 26], or treating them the context [7], where c is some desired controllable attribute(s) and $\mathbf{x}$ the generated sample. NLMs are trained as conditional language models for specific tasks that require text generation [5]. Given text sequence $\mathbf{x}_i = \{x_{i,1}, \cdots, x_{i,|\mathbf{x}_i|}\}$ and dataset $D = \{\mathbf{x}_1, \cdots, \mathbf{x}_D\}$, NLMs perform pre-training by minimizing the following likelihood under forward autoregressive factorization:

$$\mathcal{L}_{LM}(\theta) = -\sum_{i=1}^{|D|}\sum_{t=1}^{|\mathbf{x}_i|} \log P_\theta(x_{i,t}|\mathbf{x}_{i,1:t-1}), \tag{1}$$

where θ are the model parameters.

As our approach permits NLMs and PLMs to accept both control codes and free-text (text description), we call them attributes, $\mathbf{c}_{i,1:a} = \{c_1, \cdots, c_a\}$, and we can extend this equation to:

$$\mathcal{L}_{FCTG}(\theta) = -\sum_{i=1}^{|D|}\sum_{t=1}^{|\mathbf{x}_i|} \log P_\theta(x_{i,t}|\mathbf{x}_{i,1:t-1}, \mathbf{c}_{i,1:a}), \tag{2}$$

where $\mathbf{c}_{i,1:a}$ denotes the set of $|a|$ attributes. These attributes could influence the generation output, $\mathbf{x}_i$, by steering the next token distribution. Different from CTRL, CoCon, BGM, Prefix, and NRP, FCTG treats attributes as interpretation instructions rather than constraints, and detangles the conditional dependencies instead of learning them independently. To mitigate this restricted formulation, FCTG lets both $\mathbf{x}_{1:t}$ and $\mathbf{c}_{i,*}$ affect x_{t+1} in parallel at any time step, while $\mathbf{x}_{1:t}$ affects $\mathbf{c}_{i,1:a}$ in training. This paper explores how to learn these distributions and elucidates its learning framework.

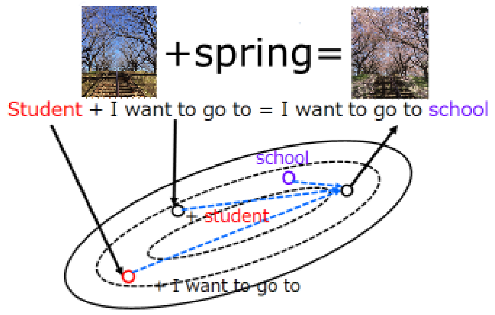

Figure 1: What is modality? (top) multi-modal search, (bottom) conditional text generation: While multi-modal search discovers entities across their differences, conditional text generation forms text by predicting the next word (in purple) from preceding words and given attributes (in red).

4 FRIENDLY CONDITIONAL TEXT GENERATOR (FCTG)

4.1 Motivation and Methodology

We explain our motivation, modality of words and attributes (e.g., review text and the name, brand, item category, customer id, or etc.), and its methodology using Figure 1. This modality requires us to project attributes, texts and words into the same space, where we can directly measure their semantic similarity. We interpret the relationship between images and texts in multi-modal search as the relationship between attributes and texts in controllable text generation. In this search domain, images and texts are expressed as embedding vectors, and are placed in the same space such that their proximity in space becomes semantic proximity. This space allows search systems to return images closest to the new vector obtained by conducting algebraic operations over these vectors (e.g., vec(image) + vec(word)). Applying this insight to controllable generation tasks, the next word could be predicted by the same operation of the previous words and their attributes, where words and attributes are also projected as vectors in the same space. Although attributes and texts reside in different regions of the semantic space, these attributes can also be seen as a text generation guide. This analogy propels us to explore how to leverage this modality in text generation and direct the generated text over attributes, and develop the Friendly Conditional Text Generator (FCTG). FCTG encodes both words and attributes, optimizes their representations in a bidirectional LM, and trains a unidirectional LM to predict the next word from preceding words and given attributes. This is why FCTG learns the attributes' continuous representations instead of calculating a distribution for each code (e.g., discrete tokens) or optimizing over them, and can accept more detailed conditions than alternatives as attributes.

4.2 Architecture of FCTG

We overview our framework in Figure 2, and define a multi-view attention mechanism (MVA) using a mixture of two different self-attention masks to capture the modality between each text (words) and their attributes while preserving the capability of PLMs. The mechanism allows FCTG to employ a bidirectional LM to encode attributes, while leveraging a unidirectional LM to generate text. We introduce MAM and ALM objectives with regard to training FCTG for the effective fusion of both LMs via MVA.

4.3 Input

FCTG integrates each text with its attributes to form input sequences, and feeds them to the Transformer. Here, we add special tokens [CLS], [EOA], [SEP], and [EOT]. [CLS] token is inserted only prior to the attributes, and denotes the class of each input unit. [EOA] token is inserted between the last attribute token and first word token. [SEP] token is assigned to the end of each sentence in each input sequence. [EOT] token is assigned only after the last token in each input sequence. The input embedding layer converts all tokens (attributes and words) into embeddings, and obtains the final representation for each token by summing these embeddings.
Attribute Embedding: We can select the tokenizer according to task or dataset, share their embedding with the linguistic part using the linguistic tokenizer, and add new tokens in this tokenizer. This learnable embedding is treated the same as word embedding via MVA. While we obtain embeddings by utilizing look-up tables as these attributes have unique embedding values for each attribute as the initial values, their final representation values are gained through training. This means that similar attributes have similar representation values, like word embeddings.
Word Embedding: Like other NLMs, FCTG tokenizes each input text as the linguistic input of token embedding, where each subword is embedded with Word Piece [49] or another NLMs/PLMs-specific tokenizer whose length equals the length of its input. Like attribute embedding, this learnable embedding can be utilized in common over texts, and their separation distance represents the closeness of their meanings.
Position/Segment Embedding: A learnable sequence position/segment embedding is added to every input element indicating its order/segment type in the input sequence, like done in other models.

4.4 Multi-view attention (MVA) and language models

As MVA aims to turn both the attributes and words into the same space, and optimize them in training, it unifies attention masks with different areas. Unlike Transformer-based models [6, 7, 28], MVA allows attributes to attend to the whole input sequence and learns their representations by switching masks. Like unidirectional LMs, FCTG embeds the input sequence, adds positional/segment encoding to each word token, and decodes the linguistic output in an auto-regressive way. We denote the embedded inputs as $\mathbf{H}_0 = [e_1, \cdots, e_{|x|}]$ of length $|x|$, and encode them into multiple levels of contextual representations $\mathbf{H}_l = Transformer(\mathbf{H}_{l-1}), l \in [1, L]$ using L-stacked Transformer blocks, where the l-th block is denoted as $\mathbf{H}_l = [h_{l,1}, \cdots, h_{l,|x|}]$ and $h_{l,|x|} \in \mathbb{R}^{d_h}$.

Inside each Transformer block, the previous layer's output $\mathbf{H}_{l-1} \in \mathbb{R}^{|x| \times d_h}$ is aggregated using multi-head self-attention, and the core of the block is multi-head attention with heads that use a causal mask to preclude attending to future tokens by using the scaled

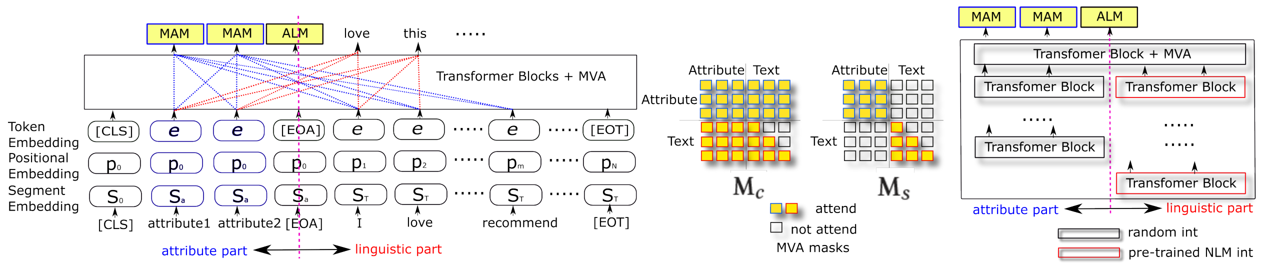

Figure 2: The architecture of FCTG: FCTG (left) consists of an attribute part encoder (bidirectional LM) with MAM and a linguistic part decoder (unidirectional LM), and integrates these parts with ALM and (center) MVA. (right) While FCTG uses masked self-attention (left-to-right) and can adopt PLMs in the linguistic part, it can employ Transformer blocks with self-attention or a simple Multi-Layer Perceptron in the attribute part. Using PLMs allows FCTG to duplicate its parameters in the linguistic (red Transformer blocks), where $Q^a/K^a/V^a$ denotes query/key/value of (solid Transformer blocks).

dot-product attention:

$$\mathbf{Q} = \mathbf{H}_{l-1}\mathbf{W}_l^Q, \mathbf{K} = \mathbf{H}_{l-1}\mathbf{W}_l^K, \mathbf{V} = \mathbf{H}_{l-1}\mathbf{W}_l^V,$$
$$Attention(\mathbf{Q},\mathbf{K},\mathbf{V}) = softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}} + \mathbf{M})\mathbf{V}, \quad (3)$$
$$M_{ij} = \begin{cases} -\infty & \text{if } i,j \text{ is the word token and } i<j, \\ 0 & \text{else} \end{cases},$$

where $\mathbf{W}_l^Q, \mathbf{W}_l^K, \mathbf{W}_l^V \in \mathbb{R}^{d_h \times d_k}$ are learnable weights for computing the queries, keys, and values, $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{|x| \times d_k}$, respectively; d_k is the shared dimensionality of the queries and keys. By applying this formulation in the attribute part, we gain $\mathbf{H}_l^A \in \mathbb{R}^{|A| \times d_h}$, while $\mathbf{H}_l^T \in \mathbb{R}^{|X| \times d_h}$ is gained in the text part, where $|A|$ and $|X|$ are the lengths of attribute and linguistic part, respectively. The attention mask, $\mathbf{M} \in \mathbb{R}^{|x| \times |x|}$, determines whether a position can attend to other positions.

To adapt PLMs to this task, it is crucial to correctly employ attributes and linguistic knowledge (tokens) from NLMs at the right time. While attributes and linguistic tokens can influence other tokens like cross-attention, they should exercise influence only with tokens of the same kind like self-attention. In order to balance this influence, our MVA unifies two different attention masks, $\mathbf{M}_c \in \mathbb{R}^{(|A|+|X|)\times(|A|+|X|)}$ and $\mathbf{M}_s \in \mathbb{R}^{(|A|+|X|)\times(|A|+|X|)}$, as illustrated in Figure 2.

This balance is automatically controlled by the gate matrix, $\mathbf{B}_c \in \mathbb{R}^{(|A|+|X|)\times(|A|+|X|)}$, and determines the relative strengths of attribute layer, $\mathbf{H}_l^A$, and linguistic layer, $\mathbf{H}_l^T$, on the top of each layer. Following Eq (3), $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ are gained from $\mathbf{H}_a \oplus \mathbf{H}_L$, so MVA can be defined as:

$$MVA(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathbf{B}_c \otimes softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}} + \mathbf{M}_c)\mathbf{V},$$
$$+ (\mathbf{1} - \mathbf{B}_c) \otimes softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}} + \mathbf{M}_s)\mathbf{V},$$
$$\mathbf{B}_c = \sigma(\mathbf{A})\mathbf{I}(\sigma(\mathbf{A}) > \mu), \mathbf{A} = [\mathbf{H}_a \oplus \mathbf{H}_L]\mathbf{W}_b + \mathbf{C}_b, \quad (4)$$
$$M_c(i,j) = \begin{cases} -\infty & \text{if } i,j \text{ is the linguistic token and } i<j, \\ 0 & \text{else} \end{cases},$$
$$M_s(i,j) = \begin{cases} -\infty & \text{if } i,j \text{ are not the same token type} \\ & \text{or if } i,j \text{ is the linguistic token and } i<j, \\ 0 & \text{else} \end{cases},$$

where $\otimes$ denotes component-wise multiplication, and $\mathbf{W}_b \in \mathbb{R}^{d_h \times (|A|+|X|)}$ and $\mathbf{C}_b \in \mathbb{R}^{(|A|+|X|)\times(|A|+|X|)}$ are learnable weights, μ is gate threshold value, and $\mathbf{I}$ is the indicator function that returns 1 if the inner statement is true and 0 otherwise. This function aims to mitigate overfitting by suppressing small values and inducing sparse activation. It applies bidirectional attention between attributes and words, where, $M_*(i,j) \in \mathbf{M}_* = 0$ allows the i-th position to attend to the j-th position and $M_{ij} = -\infty$ prevents it from attending. MVA makes each attribute token attend to all tokens in the input sequence via $\mathbf{M}_c$, and restricts each word token to attend to the word tokens in the previous position within the same sequence (left-to-right) and attribute tokens. Thus FCTG represents both attributes and words in the same space, and feeds $MVA(\mathbf{Q},\mathbf{K},\mathbf{V})$ to a feedforward layer with ReLU activation [35]. FCTG projects inputs to an inner dimension, f, with layer normalization [3] to compute $\mathbf{H}_l$ for the next layer, which generates text over arbitrary attributes.

The output of the final pointwise feed-forward layer goes through a final linear layer that acts as a classifier. As temperature-controlled stochastic sampling can generate text from trained NLMs, FCTG applies this approach to the output of MVA, $\mathbf{H}_{MVA}$, and samples the next token according to the probability:

$$\mathcal{X} = LayerNorm(\mathbf{H}_{MVA})\mathbf{W}_c, \quad p(w_i) = \frac{exp(x_i/T)}{\sum_i exp(x_i/T)}, \quad (5)$$

where $\mathbf{W}_c \in \mathbb{R}^{d_h \times V}$ is the learnable weight, V is vocabulary size, $T > 0$ is temperature, and $x_i \in \mathcal{X}$ is the score of the i-th token in the vocabulary. The next token is chosen by sampling on a multinomial distribution with probabilities clipped at the top-k tokens. While $T \longrightarrow 0$ approximates a greedy distribution, which magnifies the peaks in the probability distribution, $T \longrightarrow \infty$ flattens the distribution and makes it more uniform.

That is, MVA allows FCTG to accept new attributes as its conditions, turn them into continuous vector representations, update attribute representation by attending to other attributes and its text's representation in the training phase.

5 MODEL TRAINING

Our training consists of two tasks: Masked Attribute Modeling (MAM) and Attribute Linguistic Matching (ALM). These tasks explicitly examine the impact of attributes on modality, and ensure that the decoder focuses on both the attributes and the text (words).

5.1 Masked Attribute Modeling (MAM)

As the attribute part of FCTG is a bidirectional LM, we can optimize this part's output by applying cross-entropy loss. As an alternative objective, we propose that MAM maximize the likelihood of masked attributes given context to reconstruct these attributes. MAM makes this context include both other attributes (if any) and text in the same input sequence. We denote the linguistic input as $\mathbf{w_j} = \{\mathbf{w_{j,1}}, \cdots, \mathbf{w_{j,i}}\}$, associated attributes as $\mathbf{a_j} = \{\mathbf{a_{j,1}}, \cdots, \mathbf{a_{j,i}}\}$, and the mask index as $m \in \mathbb{N}^M$. In this training, we randomly mask out the input attributes with probability of 15%, and replace the masked attribute $\mathbf{a_{j,m}}$ with a special token [MASK], where we select input with the probability of 15% over the same attribute to avoid sample bias. FCTG is trained to predict these masked attributes using context $\mathbf{w_j}$ and other attributes $\mathbf{a_{j,\backslash m}}$, by minimizing the log-likelihood as follows:

$$\mathcal{L}_{MAM}(\zeta) = -E_{(a_{j,m},\mathbf{w_j})\sim\mathbf{D}} log P_{\zeta}(a_{j,m}|\mathbf{a_{j,\backslash m}}, \mathbf{w_j}) \tag{6}$$

where ζ indicates trainable parameters. Each pair $(a_{j,m}, \mathbf{w_j})$ is sampled from the whole training set $\mathbf{D}$. This function also employs categorical cross-entropy loss as used in other LMs.

5.2 Attribute Linguistic Matching (ALM)

As attributes and texts usually correspond to different domains, which may preclude effective cooperation without their coordination, ALM learns sequence-level alignment (rather than linguistic/attribute token-level alignment) between all attributes and the text. Given the pair of a text and corresponding attributes as input, the objective here is to predict whether the text can be semantically matched with the attributes. The first solution is to introduce constructive learning tasks, with the aim of aligning the modalities into the same space by minimizing their contrastive loss between the attribute embedding and the text embedding. While we can make each attribute and text vector, and its i-th pair, $< \mathbf{v}_{|A|,i}, \mathbf{v}_{|T|,i} >$, by computing the mean or max-over-time of all output vectors in each part (final hidden representation $\mathbf{H}_L$ or embeddings), we adopt the best, the mean of $\mathbf{H}_L^A$ and $\mathbf{H}_L^T$, in our comparison experiment. Next, we formalize the attribute embedding and the text embedding as

$$\mathcal{L}_{MAM}(\zeta) = -\sum_{b=1}^{\mathbf{B}} \sum_{i \in b} \log \frac{\exp(\mathbf{v}_{|A|,i} \cdot \mathbf{v}_{|T|,i}/\tau)}{\sum_{k \in b \backslash j} \exp(\mathbf{v}_{|A|,i} \cdot \mathbf{v}_{|T|,k}/\tau)}, \tag{7}$$

where $\mathbf{B}$ is each batch, k-th text is regarded as the negative text in a batch, $\mathbf{B}$, and τ is a temperature parameter. We can employ various texts as negative texts by constructing each batch randomly.

As an alternative, we propose a triplet objective-based function to evaluate them by similarity rather than correctness. Given anchor attribute embedding, $\mathbf{v}_a$, positive text embedding, $\mathbf{v}_p$, and negative embedding, $\mathbf{v}_n$, triplet loss tunes the model such that the distance between $\mathbf{v}_a$ and $\mathbf{v}_p$ is smaller than the distance between $\mathbf{v}_a$ and $\mathbf{v}_n$. Mathematically, this objective minimizes the following loss function:

$$\mathcal{L}_{MAM}(\zeta) = \sum_{b=1}^{\mathbf{B}} \max_{(\mathbf{v}_a,\mathbf{v}_p,\mathbf{v}_n)\sim b} (||\mathbf{v}_a - \mathbf{v}_p|| - ||\mathbf{v}_a - \mathbf{v}_n|| + \epsilon, 0), \tag{8}$$

where $\mathbf{v}_a$ and $\mathbf{v}_{p,n}$ are also obtained by averaging $\mathbf{H}_L^A$ and $\mathbf{H}_L^T$, $\mathbf{v}_p$ corresponds to $\mathbf{v}_a$, and $\mathbf{v}_n$ a different text in the same batch. $|| \bullet ||$ is a distance metric and ϵ is the margin that ensures that $\mathbf{e_p}$ is at least ϵ closer to $\mathbf{e_a}$ than $\mathbf{e_n}$. Although the sampling targets are batch units, the same policy employed in Eq (7) is used to avoid bias.

Finally, we add an alternative of this triplet objective to reduce the computation cost. This objective selects only text with minimum distance, $||\mathbf{v}_a - \mathbf{v}_n||$, as the only negative sample for i-th attribute in Eq (8).

We call the constructive, the triplet and the minimum triplet, CN, TR, and MTR, respectively, and compare them in an ablation analysis.

5.3 Optimization

We have two training regimes corresponding to the attribute-text inputs, and formulate the final pre-training objective as the sum of these objective functions (Eq (2),(6),(7/8)):

$$\mathcal{L} = \lambda_F \mathcal{L}_{FCTG}(\theta) + \lambda_M \mathcal{L}_{MAM}(\zeta) + \lambda_A \mathcal{L}_{ALM}(\zeta), \tag{9}$$

where λ_* are hyper parameters to balance the importance of the three functions. As with parameter update, we use Adaptive Moment Estimation (Adam) [18] over mini-batches, and adopt the dropout strategy [42] to optimize networks.

5.4 Fine-Tuning or Attribute-tuning

As the data used in pre-training is likely to be different from the data used in fine-tuning, and PLMs are employed, we can use the pre-trained parameters for initialization, and establish a fine-tuning process using the given data set. To project attributes, words and newly appearing words/attributes tokens in the same space, we optimize the objective function defined in Eq (9) where the evaluation uses attribute-text pairs sampled from the attribute-text pairs in this fine-tuning process.

Here, we propose attribute-tuning as an alternative to full fine-tuning for conditional generation tasks to reduce computation costs, see Figure 2. Instead of optimizing over all queries, keys, and values, $\mathbf{Q}, \mathbf{K}, \mathbf{V}$, we optimize only the parameter of the attribute part and the top layer of PLMs, $\mathbf{Q}^a, \mathbf{K}^a, \mathbf{V}^a$, as the effects of this optimization will propagate upward to all Transformer activation layers.

As shown in Figure 2, Eq (3) and (9), this approach enables us to freeze most parameters learned in PLMs, and learn only parameters $\mathbf{Q}^a, \mathbf{K}^a, \mathbf{V}^a$ in the fine-tuning phase. This not only reduces the computation cost, but allows for the easy adoption of PLMs.

6 EXPERIMENTS

6.1 Datasets and Experiment design

Datasets We conducted evaluations using the Amazon review data sets[1] and arXiv Dataset[2], as they are large publicly available datasets. As this data has multiple attributes and is suitable for evaluating models for conditional text generation, we use it in our experiments and comparisons. The corpus is balanced across stars, so each star rating constitutes 20% of the reviews in each language. For each language, there are 200,000, 5,000 and 5,000 reviews in the training, validation and test sets, respectively. The final performance comparison results are derived from the test set. We set the minimum number of reviews per reviewer and product to 20, which yielded 2,311 customer IDs and 2,381 product IDs; the other

[1]https://huggingface.co/datasets/amazon_reviews_multi
[2]https://huggingface.co/datasets/arxiv_dataset

Table 1: Basic statistics: In #attributes, each value denotes #unique customers/products (Amazon) and tokenized title (arXiv).

Dataset	#texts	#attributes	#vocabulary
Amazon	210,000	2,311+2,381	246,534
arXiv	1,506,500	25,112	565,762

reviewers and products are grouped to yield one reviewer and one product. All reviews are truncated after 2,000 characters, and all reviews are at least 50 characters long; we use only English to ease interpretation of the results. We apply the same pre-treatment to arXiv Dataset and the statistics of the resulting data set are shown in Table 1, where 25,112 words are attributes. For Amazon data, we use anonymized reviewer ID and anonymized product ID as attributes, and made the models generate reviews according to given IDs. For arXiv data, we tokenize "paper title" into words and use these words as attributes to generate "paper abstract", and made the models generate abstracts according to given tokenized title. Therefore, the attributes of Amazon and arXiv, are to be seen as set of codes and free-text, respectively.

Experimental Setup We implemented FCTG using Pytorch 1.9[3]; many parameters were set to their baseline values for fair comparison. While the attribute part of FCTG consists of 6 transformer blocks with 1280/1024 hidden size to utilize CTRL/GPT-2-medium/GPT-2-large/GPT-NeoX as the pre-trained models, with 12 attention heads, the linguistic part of FCTG places only one transformer block on top of these pre-trained models, where the weight matrix of the softmax classifier was tied to token/attribute embeddings (MAM, ALM), and λ_F, λ_M, and λ_A of Eq (9) were set to 1, 0.1, and 0,1, respectively.

As with Transformer training, we ran the models for 50 epochs using Adam with $\beta_1 = 0.9$; $\beta_2 = 0.999$ was used for optimization, over mini-batches to update parameters; the dropout strategy [42] was adopted for network optimization. The learning rate was 3e-5, with linear warmup over the first 100 steps and linear decay, where we set the dropout rate, the weight decay, the maximum length of input sequence, and the batch size to 0.1, 0.01, 50(Amazon)/200(arXiv), and 16, respectively. As with other Transformer-based models, we use publicly available models[4], and follow the published parameter settings for fair comparison. We fine-tuned all models on 8 Nvidia Tesla V100 GPUs with 256G memory.

Experiment design Our experimental objective is to investigate the following research questions.

- RQ#1 How do the results of FCTG compare to those of other models?
- RQ#2 How much do MVA, MAM, and ALM contribute to FCTG performance?
- RQ#3 Does FCTG offer controllable text generation?
- RQ#4 Does attribute-tuning lower the computation cost of FCTG training?

[3]https://pytorch.org/
[3]https://huggingface.co/transformers/pretrained_models.html
[4]https://github.com/huggingface/transformers,https://github.com/uber-research/PPLM,https://github.com/xxbidiao/plug-and-blend,https://github.com/alvin-changw/COCON_ICLR2021,https://github.com/FreddeFrallan/Non-Residual-Prompting,https://github.com/XiangLi1999/PrefixTuning

6.2 Baselines

Since our model can adopt pre-trained NLMs, it incorporates state-of-the-art NLMs, CTRL, GPT2 [39] and GPT-NeoX [1]. As the main goal of our model is to control text generation, we use the latest NLMs sharing the same goal as our baselines and omit the results of WD [41], T-CVAE [48], PPVAE [14], PPLM [9] and BGM[28] due to space constraints and show the results of GPT-2 after te fine-tuning them over the same datasets.

6.3 Evaluation Metrics

We evaluate two characteristics: whether FCTG generates text that satisfies the desired attributes and whether text quality deteriorates as we intensify attribute control.

Automated evaluation. While perplexity is a well-known automated measure of fluency, its effectiveness in open-domain text generation has been questioned [29]. As this task is a kind of language model evaluation, we use the frequently used test-set perplexity, BLEU [38], METEOR [23], ROUGE [27] using the Hugging Face Metrics[5]. The diversity of text in the passages is measured using the number of distinct n-grams (normalized by the length of text) as in Li et al. [25]. BLEU [38] is used to evaluate how many n-grams in the generated text overlap with those in the reference text. METEOR aligns the output text to the reference text and calculates sentence-level similarity scores for the alignments. ROUGE was proposed for evaluating summarization systems, and proceeds by comparing overlapping n-grams, word sequences and word pairs.

Human evaluation. We consider two types of human annotation [9]: topic and fluency testing on attribute relevance. Topic reports the fraction of samples matching the target topic (attributes) as evaluated by human annotators. Annotators were asked to evaluate the fluency of each individual sample on a scale of 1-5, with 1 being "not fluent at all" and 5 being "very fluent" as done in [21]. To be able to consistently evaluate the generated reviews or abstracts, we recruited and screened colleagues who could interpret both reviews and papers, where annotations correspond to unique attributes (i.e., user, product ID and paper title).

6.4 Text generation task (RQ#1)

For evaluating controllable text generation, we set the maximum length of generated text to 50(Amazon)/200(arXiv) for all models; only the best three models are shown in Table 2. To yield fair comparisons against the other baseline methods, we follow the settings of recent comparisons [9]. Note that the ground truth texts were excluded from training/validation data to prevent the leak of information. For most of the datasets, FCTG outperformed the baselines and achieved better performance when the number of attributes was large. These results can be explained by FCTG bridging the modality gap between attributes and words. This advance enables FCTG to learn the relationship between attributes and words, and map them in the same semantic space; the other models deal with attributes individually and derive distributions for each attribute. This is why FCTG achieved better performance than the others, where a new code (IDs) appears in the fine-tuning phase. Our representation allows multiple attributes to express contextual interdependence between attributes and words, which prevents sparsification of the

[5]https://huggingface.co/docs/datasets/how_to_metrics

Table 2: Comparison of various text generation models: In this table, *FCTG* adopts GPT-2, CTRL, and GPT-NeoX as PLMs, sets μ in Eq (4) to 0.5, and applies CE, and CN to MAM, and ALM as Eq (6,7), where we leave FCTG with GPT-NeoX out for a fair comparison. In each model, attributes (the lower row) are a pair of user ID and product ID (Amazon) and each tokenized title (arXiv; avg 11.5). Flu, PPL, and B-N denote Fluency, Perplexity, BLEU-N, respectively. The values in bold show best performance, where the bold value denotes the statistical significance for $p < 0.01$, compared to the best baseline.

Model	Amazon							arXiv						
	Flu	PPL	B-4	Meteor	Rouge-L			Flu	PPL	B-4	Meteor	Rouge-L		
					P	R	F1					P	R	F1
	↑	↓	↑	↑	↑	↑	↑	↑	↓	↑	↑	↑	↑	↑
GPT-2	3.16	16.67	0.12	0.19	0.09	0.08	0.08	3.16	7.24	0.10	0.28	0.19	0.29	0.23
Prefix [26]	3.19	16.21	0.14	0.20	0.09	0.09	0.09	3.03	7.12	0.11	0.29	0.21	0.22	0.21
NRP [6]	3.19	15.86	0.13	0.21	0.09	0.09	0.09	3.03	7.02	0.11	0.30	0.23	0.22	0.22
COCON [7]	3.17	15.93	0.13	0.21	0.09	0.09	0.09	3.03	6.88	0.11	0.29	0.24	0.23	0.23
FCTG	**3.95**	**13.83**	**0.26**	**0.28**	**0.14**	**0.13**	**0.13**	**3.78**	**2.70**	**0.14**	**0.31**	**0.31**	**0.28**	**0.29**

Table 3: Ablation analysis of FCTG on the Amazon data set: In this table, PPL denotes Perplexity. In MAM and ALM, TR/CE denote triplet/cross-entropy in Eq (6,7,8). In the columns of Rouge-L, P, R, and F are precision, recall, and F1, respectively.

Data components			Amazon						arXiv					
			PPL	B-4	Meteor	Rouge-L			PPL	B-4	Meteor	Rouge-L		
MVA	MAM	ALM				P	R	F1				P	R	F1
$\mu = 0.8$	CE	TR	14.91	0.26	0.26	0.15	0.11	0.13	3.28	0.13	0.30	0.29	0.27	0.28
$\mu = 0.5$	CE	TR	14.88	0.27	0.26	0.14	0.12	0.13	3.24	0.13	0.30	0.28	0.28	0.28
$\mu = 0$	CE	CN	14.56	0.26	0.27	0.14	0.13	0.13	3.23	0.13	0.30	0.29	0.28	0.28
$\mu = 0$	CE	TR	**13.83**	**0.26**	**0.28**	**0.14**	**0.13**	**0.13**	**2.70**	**0.14**	**0.31**	**0.31**	**0.28**	**0.29**
$\mu = 0$	CE	MTR	14.38	0.25	0.27	0.14	0.13	0.13	3.46	0.13	0.30	0.29	0.28	0.28
$\mu = 0$	w/o	TR	14.22	0.22	0.26	0.13	0.11	0.12	3.83	0.13	0.30	0.27	0.28	0.28
w/o	TR	TR	14.84	0.19	0.24	0.11	0.11	0.11	4.32	0.12	0.29	0.25	0.27	0.26

available data for learning. Hence, while variations of FCTG achieve better results than other models, we compared the best model in the benchmark with the best of these variations for each evaluation, and confirmed statistically significant benefits.

A manual error analysis showed that some instances marked as errors were in fact assessed correctly if partial matching of words in a text is allowed. In practice, failure was possible if the ground truth texts had many abbreviations or free contexts that were not syntactically correct.

6.5 Ablation analysis (RQ#2)

To investigate the individual contributions of FCTG's components to its overall performance, we conducted an ablation analysis. We removed different components and the resulting text generation quality is shown in Table 3, where both Topic and Flu show no significant differences, and are not included due to space limitations. As proposed in 5.2, we evaluated attribute-tuning in this task.

Table 3 shows that the setting with all components yields better controllable text generation across the datasets examined. To learn the modality between attributes and words, we proposed MVA, MAM and ALM as training tasks and these results confirm their effectiveness. If MVA was not used, bidirectional/left-to-right self-attention was applied to the attribute/linguistic part. These results show that the effect of MVA is so significant that it could not be replaced by the conventional self-attention strategy, where each attribute is used for conditional text generation. While MAM is introduced to discover the modality between attributes and word at the token level, ALM is designed to learn the modality between attributes and text at the input sequence level. This table shows that the addition of MAM yields better results than PPL, ALM with TR is more likely to show up in these results, but we manually confirmed its effectiveness with respect to the consistency of the generated sentences. As Transformer models need huge numbers of parameters that must be updated in fine-tuning, FCTG applies only attribute-tuning (partial update) $\mathbf{Q}^a, \mathbf{K}^a, \mathbf{V}^a$ rather than the full one, $\mathbf{Q}, \mathbf{K}, \mathbf{V}$. This attribute-tuning update strategy offers performance comparable to that achieved with full update, $\mathbf{Q}, \mathbf{K}, \mathbf{V}$. The reason why attribute-tuning has such a larger impact on accuracy than the others is that it is also directly used in MAM and ALM. This hypothesis is supported by the results achieved when MVA is excluded, i.e., attribute and text are updated independently. A comparison of update strategies reveals it's effectiveness, although it does depend on the quality and quantity of pre-training and fine-tuning data.

6.6 Case study of generated text (RQ#3)

Table 4 shows an example of the reviews generated from the Amazon dataset by the baseline models, where the bold words in each text are attribute-specific words. Since COCON uses GPT-2, COCON generates text that is similar to that produced by GPT-2. Since FCTG also uses GPT-2 and CTRL as its PLMs, FCTG also generates text similar to the text generated by these models. While differences are seen at the sentence level rather than the word level, the texts generated by FCTG include more bold words than the texts of the other models. As FCTG contains many attribute-specific words, it

Table 4: Case study of texts generated using Amazon data: We used the same product ID (PID), customer ID (CID) and seed words "I love this movie because" for fair comparison, and show the text generated by each model, where the bold emphasis denotes the top 20 most frequent words in the corresponding attributes.

GPT-2	PID:CID:[I love this movie because] I can see it as an actual movie that shows what your dreams are like.
CTRL	PID:CID:[I love this movie because] this movie in my opinion is not just an unrealistic picture I know
COCON	PID:CID:[I love this movie because] I can see it as a very real movie that shows what his work is to,
GPT-2+p	PID:CID:[I love this movie because] I can see it as a movie that shows what I expect to see and what
FCTG(+GPT-2)	PID:CID:[I love this movie because] I can see it is an **exciting** movie that shows who your **superhero** is
FCTG(+CTRL)	PID:CID:[I love this movie because] this movie has a lot of **action** and is **thrilling** in every sense and I

Table 5: Runtime comparison in fine-tuning over (upper) Amazon and (lower) arXiv: In this table, the value is the average wall time of each epoch.

CTRL	FCTG (+CTRL)	GPT2+p	NRP	COCON (+GPT2)	FCTG
36.4m	4.2m	6.8m	7.3m	9.7m	3.5m
73.2m	6.5m	8.7m	11.5m	17.2m	4.8m

captures the modalities between attributes and words and reflects them in text generation, as stated in 4.1.

6.7 Runtime (RQ#4)

While FCTG employs CTRL and GPT-2(PPLM) as its PLMs, it utilizes just their final hidden states, not the entire model. That is, we construct FCTG with fewer stacked Transformer blocks, 6, than the others (CTRL and PPLM with GPT-2) which use 48 and 24 stacked Transformer blocks), respectively. Thus it needs fewer parameters and less memory, while achieving nearly the same performance. Although the block size of FCTG appears to be small, the well-known denoising autoencoder based text generation model, BART [24], has almost the same size and achieves new state-of-the art results on question answering and summarization tasks. If the number of Transformer blocks is kept the same, FCTG converges faster with higher accuracy, although its number of parameters increases slightly ($\mathbf{M}_c, \mathbf{M}_s, \mathbf{B}_c, \mathbf{W}_b, \mathbf{C}_b$ in Eq 4)

7 DISCUSSION

The difference between a prefix word and an attribute is that the prefix word cannot refer to words appearing after it in the autoregressive language model, whereas an attribute can attend to words associated with that attribute over the whole sentence. This difference supports the validity of MVA with different masks, as shown in 6.5. Because attributes are related to the set of words rather than their order in each input sequence, the attention mask of MVA is asymmetric and the effectiveness on the conditional text generation task was confirmed in the experiments. This is why we call these attributes text generation controllers.

The effect of including modalities is to make text generation tasks easier to interpret by analogy with multi-modal search; They are made to play the role of an "information seeker" (query) and an "information provider" (key-value). As explained in 6.4, modality ensures that FCTG can map attributes and words into the same semantic space where the proximity of the distance between attributes and words reflects their semantic proximity, allowing us to directly define their similarity. By focusing on the similarities of attributes rather than on their independence, FCTG can share texts related to attributes among similar attributes, and learn the semantic relationship between attributes and words more accurately than GPT2 with prefix-tuning [26]. As shown in the ablation analysis, training FCTG through attribute-tuning reduces computation cost.

The second feature is the ability to generate text according to a consistent set of features, even when multiple features are combined or swapped. The benefit of FCTG lies in performing encoder-decoder equivalent processing with fewer parameters by applying MVA to the decoder, and this might also apply to other encoder-decoder transformer stacks [24, 40] or LSTM based models [2, 8, 32, 37, 44, 45].

Our framework can not only accept both multiple codes and free-text as conditions, but can also easily accommodate vocabulary growth. For example, customer IDs or paper author names as attributes will appear as new "words" every day, and make the vocabulary explode. In fact, even in the data sets we used, these words were not tokenized or included as intended in PLMs. FCTG tokenized them as attributes, obtained their representations via fine-tuning, and applied them to control text generation.

The problem of catastrophic forgetting is inevitable when training PLMs, but FCTG alleviates this issue by steering PLMs only at their top layer, introducing MVA with balancing weight $\mathbf{B}_c$, and attribute-tuning. Although we could not evaluate that this component and tuning directly resolved this issue, we can say that they can preserve the parameters of PLMs, adapt the target's domain knowledge to their language generation capabilities, and control text generation with regard to its attributes.

8 CONCLUSION

To improve the performance of controllable text generation, this paper proposed FCTG; it focuses on the modality of words and attributes in each text, and projects their embedding representations into the same space. This understanding led to the derivation of MVA as the mechanism, and the introduction of MAM and ALM as training models. Different from previous conditional text generation models and frameworks, FCTG can readily accept a greater variety of attributes and treat these attributes as words rather than constraints. Experiments on publicly available datasets showed that these newly proposed components help FCTG to outperform baselines in conditional text generation tasks, and generate texts that satisfy multiple attributes more easily than existing alternatives. We will extend and apply this framework to dialogue generation, prompt tuning, RAG, and image captioning tasks.

REFERENCES

[1] Alex Andonian, Quentin Anthony, Stella Biderman, Sid Black, Preetham Gali, Leo Gao, Eric Hallahan, Josh Levy-Kramer, Connor Leahy, Lucas Nestler, Kip Parker, Michael Pieler, Shivanshu Purohit, Tri Songz, Wang Phil, and Samuel Weinbach. 2021. *GPT-NeoX: Large Scale Autoregressive Language Modeling in PyTorch*. https://doi.org/10.5281/zenodo.5879544

[2] Anonymous Author(s). -. Adjustable Personalized Review Generation from Multiple Views. In *Unpublished*. -, –.

[3] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. *CoRR* abs/1607.06450 (2016).

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014).

[5] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A Neural Probabilistic Language Model. *J. Mach. Learn. Res.* 3 (Mar 2003), 1137–1155.

[6] Fredrik Carlsson, Joey Öhman, Fangyu Liu, Severine Verlinden, Joakim Nivre, and Magnus Sahlgren. 2022. Fine-Grained Controllable Text Generation Using Non-Residual Prompting. In *ACL*. 6837–6857.

[7] Alvin Chan, Yew-Soon Ong, Bill Pung, Aston Zhang, and Jie Fu. 2021. CoCon: A Self-Supervised Approach for Controlled Text Generation. In *ICLR*.

[8] Zhongxia Chen, Xiting Wang, Xing Xie, Tong Wu, Guoqing Bu, Yining Wang, and Enhong Chen. 2019. Co-Attentive Multi-Task Learning for Explainable Recommendation. In *IJCAI*. 2137–2143.

[9] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and Play Language Models: A Simple Approach to Controlled Text Generation. In *ICLR*.

[10] Cyprien de Masson d'Autume, Shakir Mohamed, Mihaela Rosca, and Jack W. Rae. 2019. Training Language GANs from Scratch. In *NeurIPS*. 4302–4313.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACLM*. 4171–4186.

[12] Adji B. Dieng, Chong Wang, Jianfeng Gao, and John William Paisley. 2016. TopicRNN: A Recurrent Neural Network with Long-Range Semantic Dependency. *CoRR* abs/1611.01702 (2016).

[13] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified Language Model Pre-training for Natural Language Understanding and Generation. In *NeurIPS*. 13042–13054.

[14] Yu Duan, Canwen Xu, Jiaxin Pei, Jialong Han, and Chenliang Li. 2020. Pre-train and Plug-in: Flexible Conditional Text Generation with Variational Auto-Encoders. In *ACL*. 253–262.

[15] Sergey Golovanov, Rauf Kurbanov, Sergey Nikolenko, Kyryl Truskovskyi, Alexander Tselousov, and Thomas Wolf. 2019. Large-Scale Transfer Learning for Natural Language Generation. In *ACL*. 6053–6058.

[16] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[17] Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A Conditional Transformer Language Model for Controllable Generation. *CoRR* abs/1909.05858 (2019).

[18] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[19] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *ICLR*.

[20] Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq R. Joty, Richard Socher, and Nazneen Fatema Rajani. 2021. GeDi: Generative Discriminator Guided Sequence Generation. In *EMNLP*. 4929–4952.

[21] Guillaume Lample, Sandeep Subramanian, Eric Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. 2019. Multiple-Attribute Text Rewriting. In *ICLR*.

[22] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, and etc. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *ICLR*.

[23] Alon Lavie and Abhaya Agarwal. 2007. METEOR: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation, WMT@ACL 2007*. 228–231.

[24] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, and et al. 2019. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *CoRR* abs/1910.13461 (2019).

[25] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A Diversity-Promoting Objective Function for Neural Conversation Models. In *NAACL*. 110–119.

[26] Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *ACL/IJCNLP*. 4582–4597.

[27] Chin-Yew Lin. 2004. Rouge: a package for automatic evaluation of summaries. In *ACL-workshop*. 25–26.

[28] Zhiyu Lin and Mark O. Riedl. 2021. Plug-and-Blend: A Framework for Plug-and-Play Controllable Story Generation with Sketches. In *AAAI*. 58–65.

[29] Chia-Wei Liu, Ryan Lowe, Iulian Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation. In *EMNLP*. 2122–2132.

[30] Xiao Liu, Da Yin, Xingjian Zhang, Kai Su, Kan Wu, Hongxia Yang, and Jie Tang. 2021. OAG-BERT: Pre-train Heterogeneous Entity-augmented Academic Language Models. *CoRR* abs/2103.02410 (2021).

[31] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR* abs/1907.11692 (2019).

[32] Chen Ma, Peng Kang, Bin Wu, Qinglong Wang, and Xue Liu. 2019. Gated Attentive-Autoencoder for Content-Aware Recommendation. In *WSDM*. 519–527.

[33] Michael Mccloskey and Neil J. Cohen. 1989. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. *The Psychology of Learning and Motivation* 24 (1989), 104–169.

[34] Fei Mi, Liangwei Chen, Mengjie Zhao, Minlie Huang, and Boi Faltings. 2020. Continual Learning for Natural Language Generation in Task-oriented Dialog Systems.. In *EMNLP*. 3461–3474.

[35] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *ICML*. 807–814.

[36] Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. 2017. Plug & Play Generative Networks: Conditional Iterative Generation of Images in Latent Space. In *CVPR*. 3510–3520.

[37] Jianmo Ni and Julian McAuley. 2018. Personalized Review Generation By Expanding Phrases and Attending on Aspect-Aware Representations. In *ACL*. 706–711.

[38] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *ACL*. 311–318.

[39] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. (2019).

[40] Colin Raffel, Noam Shazeer, Adam Roberts, and et.al. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* 21 (2020), 140:1–140:67.

[41] Abigail See, Stephen Roller, Douwe Kiela, and Jason Weston. 2019. What makes a good conversation? How controllable attributes affect human judgments. In *NAACL-HLT*. 1702–1723.

[42] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* 15, 1 (2014), 1929–1958.

[43] Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion Transformer: Flexible Sequence Generation via Insertion Operations. In *ICML*. 5976–5985.

[44] Peijie Sun, Le Wu, Kun Zhang, Yanjie Fu, Richang Hong, and Meng Wang. 2020. Dual Learning for Explainable Recommendation: Towards Unifying User Preference Prediction and Review Generation. In *WWW*. 837–847.

[45] Quoc-Tuan Truong and Hady Lauw. 2019. Multimodal Review Generation for Recommender Systems. In *WWW*. 1864–1874.

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, and etc. 2017. Attention Is All You Need. In *NIPS*. 5998–6008.

[47] Ke Wang and Xiaojun Wan. 2018. SentiGAN: Generating Sentimental Texts via Mixture Adversarial Networks. In *IJCAI-18*. 4446–4452.

[48] Tianming Wang and Xiaojun Wan. 2019. T-CVAE: Transformer-Based Conditioned Variational Autoencoder for Story Completion. In *IJCAI*. 5233–5239.

[49] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, and et. al. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR* abs/1609.08144 (2016).

[50] Peng Xu, Mostofa Patwary, Mohammad Shoeybi, Raul Puri, Pascale Fung, Anima Anandkumar, and Bryan Catanzaro. 2020. MEGATRON-CNTRL: Controllable Story Generation with External Knowledge Using Large-Scale Language Models.

[51] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *NIPS*. 5754–5764.

[52] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2018. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. In *AAAI*. 2852–2858.

[53] Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending Against Neural Fake News. In *NeurIPS*. 9051–9062.

[54] Chengkun Zhang and Junbin Gao. 2020. Hype-HAN: Hyperbolic Hierarchical Attention Network for Semantic Embedding. In *IJCAI*. 3990–3996.

[55] Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-Tuning Language Models from Human Preferences. *CoRR* abs/1909.08593 (2019).

[56] Xu Zou, Da Yin, Qingyang Zhong, Hongxia Yang, Zhilin Yang, and Jie Tang. 2021. Controllable Generation from Pre-Trained Language Models via Inverse Prompting. In *KDD*. 2450–2460.