Exploring Controllable Text Generation Techniques

Shrimai Prabhumoye, Alan W Black, Ruslan Salakhutdinov
School of Computer Science
Carnegie Mellon University
Pittsburgh PA 15213
{sprabhum, awb, rsalakhu}@cs.cmu.edu

Abstract

Neural controllable text generation is an important area gaining attention due to its plethora of applications. Although there is a large body of prior work in controllable text generation, there is no unifying theme. In this work, we provide a new schema of the pipeline of the generation process by classifying it into five modules. The control of attributes in the generation process requires modification of these modules. We present an overview of different techniques used to perform the modulation of these modules. We also provide an analysis on the advantages and disadvantages of these techniques. We further pave ways to develop new architectures based on the combination of the modules described in this paper.

1 Introduction

Controllable text generation is the task of generating natural sentences whose attributes can be controlled. The attributes to control can range from being stylistic such politeness, sentiment, formality, etc.; demographic attributes of the person writing the text such as gender, age, etc.; content such as information, keywords, entities, etc.; ordering of information, events, like plot summaries etc. Controlling various attributes of text generation has manifold applications. For instance in dialogue response generation task, work has been done in controlling persona (Zhang et al., 2018; Li et al., 2016b), controlling various aspects of the response such as politeness (Niu and Bansal, 2018), formality, authority etc, grounding the responses in external source of information (Zhou et al., 2018; Dinan et al., 2018; Ghazvininejad et al., 2018), and controlling topic sequence (Tang et al., 2019; Prabhumoye et al., 2020). Another application is story generation where you can control the ending (Peng et al., 2018), the persona (Chandu et al., 2019), the plot (Yao et al., 2019), and the topic sequence (Huang et al., 2019). Controllable text generation is also used to modulate the formality and politeness of emails (Madaan et al., 2020). Report generation can be controlled by pulling disparate source documents into a coherent unified whole, which can use a shared set of sources such as Wikipedia article generation (Liu et al., 2018; Prabhumoye et al., 2019).

Although there is a large body of prior work in controllable text generation, there is no unifying theme. Each work addresses a specific task in a specific context. In this paper we outline a new schema which connects prior work and provides an insight into various aspects of controllable text generation. The schema contains five modules that cover the overall generation pipeline and provide an understanding of the effect of each component on the generation process. Prior work has focused on specific parts of the schema that we outline here and we provide insights into their similarities. We provide an overview of these modules and also present an exploration of the various techniques used to control and update each of these modules.

Most of the controllable text generation tasks can be framed as conditional language generation tasks. They have an input or a *source* sequence $\mathbf{U}$ and an output or a *target* sequence $\mathbf{Y}$ to be generated. In this case, we model the probability of the *target* sequence conditioned on the *source* sequence given by $P(\mathbf{Y}|\mathbf{U}) = \prod_{t=1}^{T} P(\mathbf{y}_t|\mathbf{U}, \mathbf{y}_{<t})$. The generation of the target tokens of the sequence $\mathbf{Y}$ unfolds as a time series where each token $\mathbf{y}_t$ is generated at a time step $\mathbf{t}$. At a given time step t, a generative model takes
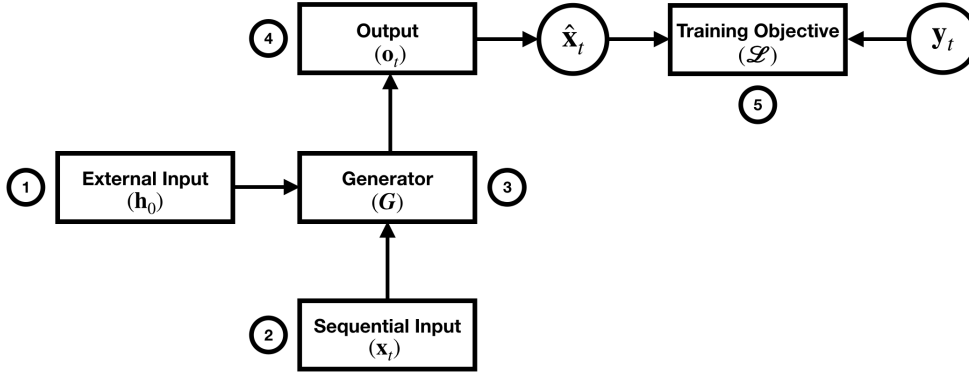
Figure 1: Modules that control the generation process. Each module is numbered by the circle next to it.

in the previous hidden state $\mathbf{h}_{t-1}$ and the input $\mathbf{x}_t$ at current time step. It performs a set of operations denoted by G to produce the output $\mathbf{o}_t$ which is used to predict token $\hat{\mathbf{x}}_t$. The ground truth token to be generated is denoted by $\mathbf{y}_t$.

Figure 1 shows the schema proposed in this work consisting of five modules which can be used for controlling the generation process: (1) **External Input** module is responsible for the initialization $\mathbf{h}_0$, of the generation process. (2) **Sequential Input** module is the input $\mathbf{x}_t$ at each time step of the generation. (3) **Generator Operations** module performs consistent operations or calculations on all the input at each time step. (4) **Output** module is the output $\mathbf{o}_t$ which is further projected on to the vocabulary space to predict the token $\hat{\mathbf{x}}_t$ at each time step. (5) **Training Objective** module takes care of the loss functions used for training the generator.

This schema provides an insight into the contributions of the various modules for controllable text generation. The main advantage of this schema is that it can be used with any algorithmic paradigm like sequence-to-sequence, adversarial methods, reinforcement learning, etc. The schema can also be used with non-autoregressive algorithms which may generate text using graphical structures like trees (Welleck et al., 2019; Guo et al., 2019). In this paper, we focus on how this schema can be used to describe controllable text generation focusing particularly on the use of autoregressive models. This work paves way to designing new architectures based on our schema. This can be done by identifying promising techniques for each module and then combining them. Our schema can also be potentially used for applying these techniques on new tasks of similar nature. It also provides a standardized framework to position and compare new architectures with existing techniques.

The prior work on unifying text generation models has mostly focused on building efficient tool-kits and modular views of generation. For instance, (Reiter and Dale, 2000) details seven sub-tasks which are conceptually distinct to describe the generation process. These sub-tasks can be modelled separately or in some cases they may interleave. In (Reiter and Dale, 2000), these seven sub-tasks are primarily characterized as content or structure tasks. Note that Reiter and Dale (2000) is not specific to neural text generation. Our work focuses specifically on controlling attributes in neural text generation process. We don't divide the generation pipeline into several sub-tasks but we divide the neural text generation process into modules all of which are required for generation. In (Hu et al., 2019b), the focus is on building a toolkit for various text generation tasks based on the three properties of versatility, modularity and extensibility. This work enlists few model architectures and learning paradigms for various text generation tasks. In our work, we focus only on the generation process of controllable text generation tasks. We specifically detail the inputs, outputs and operations of the generation process. We do not provide any specific examples of architectures but provide an overview of the basic underlying modules which can be used with any learning paradigm. Xie (2017) provides a practical guide to the neural generation process describing it in terms of initialization, optimization, regularization and decoding strategies. Our work on the other hand does not delve into the implementation details of the generation pipeline but provides an overall schema for understanding of the various components involved.

In the remainder of the paper, we denote the representation of the control attribute by $\mathbf{s}$ and the representation of the input or *source* sentence returned by the encoder as $\mathbf{h}_e$. In what follows, we first describe the possible ways of controlling attributes by modulating the *external input* in §2, the *sequential input* in §3, the *generator operations* in §4, the *output* in §5 and the *training objective* in §6. At the end of each section, we provide an analysis of each of the techniques described and how they fit together.

2   External Input

In this section we discuss the different techniques which can be used to control the generation process by updating the initialization of the generator $\mathbf{h}_0$. In the standard generation process, $\mathbf{h}_0$ is equal to $\mathbf{h}_e$. This is marked as module (1) in Figure 1.

2.1   Arithmetic or Linear Transform

One of the easiest ways to control the generation is to concatenate a control vector $\mathbf{s}$ to output of the encoder $\mathbf{h}_e$. The external input of the decoder $\mathbf{h}_0$ will be $[\mathbf{h}_e; \mathbf{s}]$, where $[a; b]$ denotes concatenation. Here, the control vector $\mathbf{s}$ would provide the generator with a strong signal to guide the generation process.

Fu et al. (2018) use this technique to control the style representation for their generator. The encoder builds representation that is devoid of the style and only retains content. The control vector for style is then concatenated to the encoder representation to initialize the decoder. This technique is commonly used in (Ghazvininejad et al., 2018; Zhou et al., 2018; Dinan et al., 2018) to concatenate information from external sources to dialogue context to generate dialogue responses. Chandu et al. (2019) concatenate personality representation $\mathcal{P}$ derived from a separate corpus to generate visual stories. They also experiment with a simple arithmetic operation on $\mathbf{h}_e$ given by $\mathbf{h}_0 = \mathbf{h}_e - \mathcal{S} + \mathcal{P}$ to get the initialization of the generator (here $\mathcal{S}$ denotes the average representation of the story). They observed that while concatenation technique is better at preserving the meaning of the generated story, the arithmetic operation provides a better signal of the personality for the generation process.

Hoang et al. (2016) uses both the concatenation technique as well as performs a linear transform of $\mathbf{s}$ to obtain $\mathbf{h}_0$ for language modelling task. The control vectors in this case represents meta data such as key-words, topics etc. In case of the linear transform $\mathbf{h}_0 = \texttt{tanh}(\mathbf{W}_1 \mathbf{h}_e + \mathbf{W}_2 \mathbf{s} + \mathbf{b})$. The paper also explores adding the control vector to the encoder representation ($\mathbf{h}_0 = \mathbf{h}_e + \mathbf{s}$).

In case of addition, the resulting $\mathbf{h}_0$ would be averaged representation of the input representation $\mathbf{h}_e$ and $\mathbf{s}$. Information could be lost in this case as control is not explicit. In case of concatenation, if the size of the control vector $\mathbf{s}$ is too small compared to $\mathbf{h}_e$, then $\mathbf{s}$ can be over-shadowed by $\mathbf{h}_e$ and the generator may not be able to pay attention to $\mathbf{s}$. Hence it is important to choose comparable dimensions for $\mathbf{s}$ and $\mathbf{h}_e$. But this increases the size of model considerably and could be quite costly. Linear transform avoids these issues and performs better than the other two techniques for Hoang et al. (2016).

2.2   Stochastic Changes

Kingma and Welling (2014) introduce variational auto-encoder, where you can stochastically draw a continuous latent variable $\mathbf{z}$ from a Gaussian distribution. The initialization of the generator $\mathbf{h}_0$ is based on this latent variable. Bowman et al. (2016) use this concept for generating sentences from this continuous latent representation. This process of changing the encoder state $\mathbf{h}_e$ can only be used with Kullback-Leibler (KL) Divergence training objective described in §6.2.

In (Wang et al., 2019b), Variational Auto-Encoder (VAE) is used to guide the generation process with topics of a document. A gaussian mixture model is used to incorporate topics into latent variables. In (Xu et al., 2019), VAE is used to control for sentiment attribute in style transfer task by constraining the posterior mean to a learned probability simplex.

Such a design of controllable text generation works when the control attributes can be represented as latent variables for example style, topics, strategies etc. This design is difficult to work for content grounded text generation tasks where specific information, keywords or entities have to guide the generation process.

2.3 Decompose

The encoder representation $\mathbf{h}_e$ can be decomposed into multiple subspaces, each of which signifies a different attribute to be controlled. Liu and Lapata (2018) split the encoder representation $\mathbf{h}_e$ into two components, one which represents the structure in the document and the other represents the semantic information. This formulation was used by (Balachandran et al., 2020) for controlling structure in abstractive summarization. This work performs the split with respect to the dimensions of $\mathbf{h}_e$. The method forces the first n dimensions of $\mathbf{h}_e$ to capture meaning and the latter to capture structure. Balachandran et al. (2020) also show quantitative and qualitative analysis on the types of structures of documents learnt by this technique.

Romanov et al. (2019) decompose the encoder representation $\mathbf{h}_e$ into a form vector $\mathbf{f}$ and a meaning vector $\mathbf{m}$. During the training phase, a *discriminator* enforces $\mathbf{m}$ to not carry any information about the form using an adversarial loss and a *motivator* is used for a motivational loss that encourages $\mathbf{f}$ to carry the information about the form. The generation process can then be guided to adhere to the desired target form. As opposed to splitting $\mathbf{h}_e$ with respect to dimensions, this work learns subspaces $\mathbf{W_m}$ and $\mathbf{W_f}$ given by $\mathbf{m} = \texttt{tanh}(\mathbf{W}_m\mathbf{h}_e + \mathbf{b}_m)$ and $\mathbf{f} = \texttt{tanh}(\mathbf{W}_f\mathbf{h}_e + \mathbf{b}_f)$ respectively. When $\mathbf{h}_e$ is projected on $\mathbf{W}_m$, it yields the meaning vector $\mathbf{m}$ and similarly when it is projected on $\mathbf{W}_f$ it yields the form vector $\mathbf{f}$. This work shows qualitatively how $\mathbf{m}$ and $\mathbf{f}$ are learnt in the subspaces using t-SNE plots. It also shows quantitatively the use of $\mathbf{m}$ and $\mathbf{f}$ in downstream paraphrase detection tasks. This builds interpretable representations for control attributes. Although, the effectiveness of this technique is not yet proven in the style transfer task or the abstractive summarization task. In both the above mentioned works, the models learns interpretable representations of control attributes but were not able to beat state of the art methods in their respective tasks. It is also worth noting that learning good decomposed vectors is especially hard when no supervision is provided on what the decomposed components are supposed to learn.

This technique works well when the representation space of the input $\mathbf{x}$ can be decomposed into subspaces which can represent the control attributes. This means that the input $\mathbf{x}$ needs to contain signal of the control attributes. It is unlikely to work when the control attributes need to be externally provided. For example in case of content grounded generation tasks described in (Prabhumoye et al., 2019; Dinan et al., 2018; Zhou et al., 2018), the input may not necessarily contain the content that needs to be generated. A separate input of the content to be generated is provided in these cases.

2.4 External Feedback

A regularizer is often used to control the external input $\mathbf{h}_0$ to the generator. In many cases, an adversarial loss to manipulate the latent space is used as an external feedback mechanism. This essentially controls the latent space of the encoder which is eventually provided as an initialization to the generator. In (Fu et al., 2018), a multi-layer perceptron (MLP) is used for predicting the style labels from $\mathbf{h}_0$. Similarly, the adversarial loss is also used in (Wang et al., 2019a) to control the latent representation $\mathbf{h}_0$ for style attributes. In (Romanov et al., 2019), an adversarial loss is used to ensure that the meaning representation $\mathbf{m}$ does not carry any style signals. The adversarial loss is obtained by training a discriminator which takes as input a representation $\mathbf{m}$ and indicates if it carries the target style signal. Similarly, this work also employs a motivator loss which is the opposite of the adversarial loss to ensure that the style representation $\mathbf{f}$ actually does carry the stylistic information. John et al. (2019) use multiple losses to control the style and content information represented in $\mathbf{h}_0$.

The discriminator which provides external feedback has to be jointly trained with the generator. This technique can be useful with the decompose technique to ensure that the decomposed sub-spaces represent the desired control attributes.

3 Sequential Input

In this section we discuss the different techniques which can be used to manipulate the sequential input $\mathbf{x}_t$ to the decoder at each time step. $\mathbf{x}_t$ here is used to denote the word embedding of the token at time step t. This is marked as position (2) in Figure 1.

3.1 Arithmetic or Linear Transform

Similar to changing the initialization, we can change the input to the decoder by concatenating the information at each time step with some additional control vector $\mathbf{s}$. Typically, teacher forcing method (Williams and Zipser, 1989) is used to train the generator. At time step t, the generator takes as input the word embedding $\mathbf{x}_t$ of the word that was predicted at step $t-1$ and predicts the word to be generated $\mathbf{y}_t$ at the current time step. Note that $\mathbf{x}_t = \mathbf{y}_{t-1}$. The input $\mathbf{x}_t$ can be concatenated with $\mathbf{s}$ at each time step to control the generation process. Hence, $\tilde{\mathbf{x}}_t = [\mathbf{x}_t; \mathbf{s}]$.

Noraset et al. (2017), use this technique in the task of definition modeling. They concatenate word embedding vector $\mathbf{s}$ of the word to be defined at each time step of the definition generation process. Unfortunately, for this task, this technique has not proved to be effective compared to other techniques of controlling the generation. Zhou et al. (2018) concatenate the hidden representation of the external source of information $\mathbf{s}$ to each time step of dialogue response generation. Similarly, Prabhumoye et al. (2019) also concatenate the hidden representation of the external source of information $\mathbf{s}$ to each time step of Wikipedia update generation process. This technique did not achieve impressive results in this work as well. Harrison et al. (2019) concatenate a side constraint $\mathbf{s}$ which represents style and personality into the generation process. For this task of generating language from meaning representations with stylistic variation, this method performed better than conditioning the encoder with side constraint in terms of BLEU metric. Chandu et al. (2019) also concatenate the personality representation $\mathcal{P}$ at each time step of the story generation process. This is used to control the personality of the visual stories. In addition to concatenation, this work proposes to modify the sequential input as $\tilde{\mathbf{x}}_t = \mathbf{x}_t - \mathcal{S} + \mathcal{P}$ (here $\mathcal{S}$ denotes the average representation of the story and $\mathcal{P}$ denotes the representation of the personality). The latter technique is better at generating personality conditioned stories than the concatenation technique. Neither of these techniques prove to be conclusively better than making similar changes to the external input module (§2.1). Note that in this technique, changes are made directly to the input of generation and not the context which is the case with external input. Also, most of the prior work has focused on recurrent neural network and its variants for making such changes. It could be interesting to see such changes made to transformers (Vaswani et al., 2017).

4 Generator Operations

This module takes in the external input $\mathbf{h}_0$, the sequential input $\mathbf{x}_t$ at time step t and performs the same set of computations (G) to return an output $\mathbf{o}_t$. Changes can be made to the set of operations G to include the the control vector $\mathbf{s}$ in computing $\mathbf{o}_t$. This is shown as position (3) in Figure 1.

4.1 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are designed to model sequential information. RNNs perform the same operations for every element of a sequence, with the output depending on previous computations. This recurrence serves as a form of memory. It allows contextual information to flow through the network so that relevant outputs from previous time steps can be applied to network operations at the current time step. Theoretically, RNNs can make use of information in arbitrarily long sequences, but empirically, they are limited to looking back only a few steps.

The Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) units are a type of RNNs that have additional 'memory cell' apart from standard units of basic RNNs. The memory cell can maintain information in memory for long periods of time. A set of gates is used to control when information enters the memory, when it's output, and when it's forgotten. This architecture lets them learn longer-term dependencies. The vanishing gradient problem of RNNs is resolved here. Gated Recurrent Units (GRUs) (Cho et al., 2014) are similar to LSTMs, but use a simplified structure designed to adaptively capture dependencies of different time scales. They also use a set of gates to control the flow of information, but they don't use separate memory cells, and they use fewer gates.

The computations of the RNN or its variants can be modified to account for the control attribute. Additional gates can be added or the control attribute can be provided as an additional input to the standard gates of RNNs. Gan et al. (2017) propose a variant of the LSTM model, named factored LSTM, which

controls style representation in image caption task. The parameters of the LSTM module which are responsible to transform the input $\mathbf{x}_t$ are factored into three components $\mathbf{U}$, $\mathbf{S}$ and $\mathbf{V}$. The operations of the input ($\mathbf{i}_t$), forget ($\mathbf{f}_t$) and output gate ($\mathbf{o}_t$) are given by:

$$
\begin{aligned}
\mathbf{i}_t &= \texttt{sigmoid}(\mathbf{U}_{ix}\mathbf{S}_{ix}\mathbf{V}_{ix}\mathbf{x}_t + \mathbf{W}_{ih}\mathbf{h}_{t-1}) \\
\mathbf{f}_t &= \texttt{sigmoid}(\mathbf{U}_{fx}\mathbf{S}_{fx}\mathbf{V}_{fx}\mathbf{x}_t + \mathbf{W}_{fh}\mathbf{h}_{t-1}) \\
\mathbf{o}_t &= \texttt{sigmoid}(\mathbf{U}_{ox}\mathbf{S}_{ox}\mathbf{V}_{ox}\mathbf{x}_t + \mathbf{W}_{oh}\mathbf{h}_{t-1}) \\
\tilde{\mathbf{c}}_t &= \texttt{tanh}(\mathbf{U}_{cx}\mathbf{S}_{cx}\mathbf{V}_{cx}\mathbf{x}_t + \mathbf{W}_{ch}\mathbf{h}_{t-1})
\end{aligned}
$$

Particularly, the matrix set $\{\mathbf{S}\}$ is specific to each style in the task and is responsible to capture the underlying style features in the data.

In (Kiddon et al., 2016), the GRU unit is modified to accommodate extra inputs - goal $\mathbf{g}$ and agenda items E_t^{new} in the recipe generation task. The operation of the new component $\tilde{\mathbf{h}}_t$ is given by:

$$
\tilde{\mathbf{h}}_t = \texttt{tanh}(\mathbf{W}_h\mathbf{x}_t + \mathbf{r}_t \odot \mathbf{U}_h\mathbf{h}_{t-1} + \mathbf{s}_t \odot \mathbf{Y}\mathbf{g} + \mathbf{q}_t \odot (\mathbf{1}_L^T\mathbf{Z}\mathbf{E}_t^{new})^T)
$$

where $\mathbf{s}_t$ is a goal select gate and $\mathbf{q}_t$ is a item select gate. With this modification, the generation process is controlled for the items to be generation in the recipe and the goal.

Wen et al. (2015) adapt the LSTM to control the dialogue act information in the generation process. The operation to compute the cell value $\mathbf{c}_t$ is given by:

$$
\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t + \texttt{tanh}(\mathbf{W}_d\mathbf{d}_t)
$$

The dialogue act representation $\mathbf{d}_t$ is build using another LSTM cell.

RNNs, LSTMs and GRUs are commonly used to model controllable text generation tasks (Prabhumoye et al., 2019; Rao and Tetreault, 2018; See et al., 2017; Zhou et al., 2018; Fu et al., 2018). Most of these variants still have trouble remembering long sequences and are hence commonly used with attention mechanism (§5.1) on the source sequence.

4.2 Transformer

Transformers are proposed by (Vaswani et al., 2017) and they rely on attention mechanism to draw global dependencies between input and output. The Transformer uses stacked self-attention and point-wise, fully connected layers for both the encoder and decoder. The encoder stacks N identical layers, each of which has two sub-layers. The first sub-layer is a multi-head self-attention mechanism (§5.1), and the second sub-layer is a positionwise fully connected feed-forward network. Each sub-layer uses residual connections around each of the sub-layers, followed by layer normalization. The decoder has an additional third sub-layer, which performs multi-head attention over the output of the encoder stack.

Since, attention mechanism is at the core of this generator, the decoder can attend over all positions of input sequence. Computations over a sequence can be parallelized in this case and hence it is faster. The modifications made to the computing units of RNN mentioned in §4.1 which use parameters specific to control attributes such as style, dialog act etc. have not been explored with the transformers architecture.

4.3 Pre-trained models

Recently pre-trained conditional language models are used for text generation like GPT (Radford et al., 2018), GPT2 (Radford et al., 2019), XLNet (Yang et al., 2019), etc. Several works have fine-tuned the pre-trained models for downstream controllable text generation tasks (Sudhakar et al., 2019; Dinan et al., 2018; Urbanek et al., 2019). The language modeling aspects of generation like fluency and grammaticality are already learnt if pre-trained models are used.

These models are hard to fine-tune for sequence-to-sequence tasks such as machine translation, abstractive summarization etc. BART (Lewis et al., 2019) is a denoising autoencoder built with a sequence-to-sequence model and is particularly effective when fine tuned for text generation. Alternatively, T5 (Raffel et al., 2019) treats every NLP problem as a "text-to-text" problem, i.e. taking text as input and producing new text as output. Hence, it can be adapted to controllable text generation tasks. Dathathri et al. (2019)

propose a Plug and Play Language Model (PPLM) for controllable language generation. It combines a pre-trained LM with one or more simple attribute classifiers that guide text generation without any further training of the LM. This is similar to the classifier feedback technique described in §6.3. Some of the other techniques described in this paper such as stochastic changes §2.2 , external feedback §2.4 and §5.2, decompose §2.3 etc would be hard to incorporate into pre-trained language models without modifying the model architecture or fine-tuning entailing the significant cost of retraining.

5 Output

In the standard generation process, $\mathbf{o}_t$ is the output of the generator module which is projected to the vocabulary space to predict the token $\hat{\mathbf{x}}_t$. Here, we discuss the various techniques used to modulate the sequential output $\mathbf{o}_t$ at each time step t, before projecting it to the vocabulary space. This is marked as position (4) in Figure 1.

5.1 Attention

Attention is the most popular way of guiding the generation process. It is typically used to guide the generation process to focus on the source sequence (Bahdanau et al., 2015). The attention calculating module takes as input the current hidden state $\mathbf{h}_t$ of the generator at each time step $\mathbf{t}$. The aim of this module is to determine a context vector $\mathbf{c}_t$ that captures relevant source-side information to help predict the token $\hat{\mathbf{x}}_t$. In case of *global attention*, all the hidden states of the encoder are considered to calculate the context vector $\mathbf{c}_t$ (Luong et al., 2015). This faces the the downside of expensive calculation especially for longer source sequences like documents. To overcome this challenge, *local attention* only chooses to focus only on a small subset of the source positions per target word. In this case, $\mathbf{c}_t$ is calculated over a window of size D of the source hidden states.

Vaswani et al. (2017) view attention as a mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key. This work proposes the simultaneous use of *scaled dot-product* attention which helps in parallelizing computation and a *multi-headed* attention which allows the model to jointly attend to information from different representation subspaces at different positions.

Sudhakar et al. (2019) use self-attention to control for style by simply adding a special target style token in the source sequence. Dinan et al. (2018) also use transformers to attend over information from external document for guided dialogue response generation. (Zhang et al., 2018) uses the encoded representation of personas to compute the attention weights $\mathbf{a}_t$ at a given time step of the decoder. The attention is re-weighted according to the persona of the response to be generated in dialogue. So far, work has not been done to modulate the attention weights to control for attributes like style, topic, content etc.

5.2 External Feedback

The output latent space of the generator can be controlled by external feedback. Similar to changing the external input $\mathbf{h}_0$, the output latent space can also be changed using adversarial loss. In (Logeswaran et al., 2018), an adversarial loss is used which encourages the generation realistic and attribute compatible sentences. The adversarial loss tries to match the distribution of sentence and attribute vector pairs $(\mathbf{x}, \mathbf{s})$ where the sentence can either be a real or generated sentence. Similarly, in (Shen et al., 2017), a two discriminator losses in the style transfer task. Each discriminator is trained to distinguish between a sentence which came from the real target attribute distribution and a sentence that was transferred from source to target attribute. This work uses Professor-Forcing (Lamb et al., 2016) to match the hidden states of the generator and the discriminator. Gong et al. (2019) also control the output latent space by providing different types of rewards like style reward, semantic reward and fluency reward in the reinforcement learning setup. The discriminator used to obtain the adversarial loss has to be jointly trained with the generator.

5.3 Arithmetic or Linear Transform

Hoang et al. (2016) demonstrate three simple ways of changing the output $\mathbf{o}_t$ of an RNN to control for meta information like topic, keywords etc. The three ways demonstrated in (Hoang et al., 2016) are: (1) addition, where the modified output $\tilde{\mathbf{o}}_t$ is $\tilde{\mathbf{o}}_t = \mathbf{o}_t + \mathbf{s}$, (2) concatenation, where the modified output $\mathbf{o}_t$ ($\tilde{\mathbf{o}}_t = [\mathbf{o}_t; \mathbf{s}]$), and (3) using a perceptron layer dependent on $\mathbf{s}$ and $\mathbf{o}_t$. In this case, $\tilde{\mathbf{o}}_t$ is given by $\tilde{\mathbf{o}}_t = \texttt{tanh}(\mathbf{W}_o\mathbf{o}_t + \mathbf{W}_s\mathbf{s} + \mathbf{b}_o)$. In each of the three cases, the modified output $\tilde{\mathbf{o}}_t$ is then projected to the vocabulary space to predict the token $\hat{\mathbf{x}}_t$.

6 Training Objective

In this section we describe various methods used to control the generation using objective functions. The output $\mathbf{o}_t$ at each time step t of the generation process is projected to the vocabulary space using a linear transform ($\hat{\mathbf{o}}_t = \mathbf{W}_o\mathbf{o}_t + \mathbf{b}$). A token $\hat{\mathbf{x}}_t$ is predicted from the vocabulary by passing $\hat{\mathbf{o}}_t$ through a softmax function and taking the max value. The predicted token $\hat{\mathbf{x}}_t$ is compared with the reference token $\mathbf{y}_t$ using a loss function. This loss function can be tweaked to ensure that the generated text carries the desired control attributes.

6.1 General Loss Objective

Here, we describe the loss objectives commonly used in natural language generation tasks. These loss objectives do not try to control for any attribute. Instead they try to ensure fluent, grammatical and diverse generations.

Cross Entropy Loss: This is the basic loss used to compare the generated tokens with the reference tokens and is used in all text generation process. At each time step t, the generation has to predict a token from the vocabulary. Hence, it could be seen as a classification problem with number of classes being equal to vocabulary size. The categorical cross entropy loss is given by $-\Sigma_{c=1}^{M}\mathbf{y}_{t,c}\texttt{log}(p_{t,c})$. Here $p_{t,c}$ is the probability of the token c at time step t. Note that $p_t = \texttt{softmax}(\tilde{\mathbf{o}}_t)$ is the probability distribution over the vocabulary.

Unlikelihood loss: This maintains a set of negative candidates which is based on repeating tokens or n-grams and frequent tokens (Welleck et al., 2020). This set is updated at each time step as tokens are generated. This works at both token and sequence level and the objective tries to minimize the repetitions in generations. This is used at train time in augmentation with the maximum likelihood objective and can be used for any task.

Decoding strategies: These strategies are not used as a loss objective during training. Many of these objectives rely on post-hoc decoding strategies such as stochastic decoding which include Top k-sampling (Fan et al., 2018), nucleus sampling (Holtzman et al., 2020), or beam search variants (Paulus et al., 2018; Kulikov et al., 2019; Vijayakumar et al., 2018; Holtzman et al., 2018).

Specifically, we discuss the Diversity-Promoting objective which is used to generate a varied set of sentences given similar inputs. Particularly, Li et al. (2016a) use Maximum Mutual Information (MMI) as an objective function for the dialogue response generation task. Most generation systems use maximum likelihood objective but this objective additionally tries to reduce the proportion of generic responses. It is given by:

$$\hat{\mathbf{T}} = \texttt{argmax}_T\{\texttt{log}p(\mathbf{T}|\mathbf{S}) - \lambda\texttt{log}p(\mathbf{T})\}$$

where $\hat{\mathbf{T}}$ is the generated target sequence, $\mathbf{T}$ is the reference target sequence and $\mathbf{S}$ is the source sequence. The second term controls the generation of the high frequency or the generic target sequences. Note that this objective is only used during the inference and the generators are trained using cross entropy loss. Zhang et al. (2018), also use a diversity encouraging objective for dialogue response generation. They train a discriminator to calculate similarity between the source $\mathbf{S}$ and target $\mathbf{T}$ ($D_\psi(\mathbf{T}, \mathbf{S})$), as well as between the source $\mathbf{S}$ and the generated target $\hat{\mathbf{T}}$ ($D_\psi(\hat{\mathbf{T}}, \mathbf{S})$). They finally try to minimize the difference between $D_\psi(\mathbf{T}, \mathbf{S})$ and $D_\psi(\hat{\mathbf{T}}, \mathbf{S})$.

6.2 KL Divergence

The Kullback-Leibler (KL) Divergence score, quantifies how much one probability distribution differs from another probability distribution. The KL divergence between two distributions $\mathcal{Q}$ and $\mathcal{P}$ is often stated using the notation $\text{KL}(\mathcal{P} \parallel \mathcal{Q})$, where the operator "$\parallel$" indicates *divergence* or $\mathcal{P}$'s divergence from $\mathcal{Q}$. Note that KL Divergence is not symmetric i.e $\text{KL}(\mathcal{P} \parallel \mathcal{Q}) \neq \text{KL}(\mathcal{Q} \parallel \mathcal{P})$. KL divergence can be used to minimize the information loss while approximating a distribution. In text generation, the KL Divergence is combined with the evidence lower bound (ELBO) to approximately maximize the marginal likelihood of data $p(\mathbf{x})$ which helps in better generations. This objective is used in variational autoencoders and its variants in combination with sampling techniques described in §2.2. This objective fits in the controllable text generation paradigm because it allows you to approximate the posterior distribution of the control variables in the latent $\mathbf{z}$-space.

6.3 Classifier Loss

This loss is specifically used to ensure that the generated tokens $\hat{\mathbf{x}}$ comply with the control attributes $\mathbf{s}$. Note the difference between this loss and the external feedback loss used for the *external input* module and the *output* module is that this loss operates at the token level and the external feedback loss works on the latent hidden representations.

In case of style transfer task, this loss is used to guide the generation process to output the target style tokens. Some works (Prabhumoye et al., 2018; Sudhakar et al., 2019; Hu et al., 2017) use this loss to discriminate between all the styles in their task (one verses all fashion). This type of design will suffer from low accuracy scores when the number of styles increases. To counter this problem, this loss can be setup to calculate if the generated sentence $\hat{\mathbf{x}}$ belongs to style $\mathbf{s_1}$ or not and similarly to calculate another separate loss term for each style (Chandu et al., 2019). This type of loss design encounters increasing number of loss terms depending on the number of styles. The third way to motivate this loss term is to discriminating between a sentence $\mathbf{x}$ from data which belongs to style $\mathbf{s_1}$ and a generated sentence $\hat{\mathbf{x}}$ which belongs to the same style $\mathbf{s_1}$ (Yang et al., 2018). Again, you would need as many loss terms as the number of styles in this case. All of these works use cross entropy loss function to measure their losses.

Hu et al. (2019a) use a classifier based loss in the visual storytelling task. The classifier is a pre-trained language model (Devlin et al., 2019) used to measure the coherence between generated sentences of the story. Particularly, the classifier takes as input two sentences at a time $\hat{\mathbf{x}}_1$ and $\hat{\mathbf{x}}_2$ and outputs a binary label which indicates if $\hat{\mathbf{x}}_2$ follows $\hat{\mathbf{x}}_1$. In this case, the control variable is coherence in stories which is used to guide the generator to produce consistent sentences.

6.4 Task Specific Loss

Depending on the end task and the attribute to be controlled, you can design different loss objectives to ensure that generations abide by the target attributes.

Strategy Loss: Zhou et al. (2020) use a dialogue strategy based objective to generate responses for negotiation tasks. This task has ground truth strategies that lead to better negotiations. This loss captures the probability of a particular strategy occurring for the next utterance given the dialogue history. It guides the generator to align the responses with particular strategies.

Coverage Loss: Generating repeated words or phrases is a common problem for text generation systems, and this becomes especially pronounced for multi-sentence text generation task such as abstractive document summarization. See et al. (2017) introduce a *coverage loss* which penalizes repeatedly attending to the same locations of the source document.

Structure loss: Li et al. (2018) introduce two new loss objectives *structural compression* and *structural coverage* based on sentence-level attention. These objectives are specially designed for the task of abstractive document summarization. *structural compression* is used to generate a sentence by compressing several specific source sentences and *structural coverage* is used to cover more salient information of the original document. These objectives leverage document structure in document summarization, and

explore the effectiveness of capturing structural properties of document summarization by regularization of the generative model to generate more informative and concise summaries.

7 Discussion

Discrete space issues: The classifier loss (§6.3) is used to determine if the generated tokens $\hat{\mathbf{x}}$ are in accordance with the target control attribute $\mathbf{s}$. To calculate the loss, the generated tokens $\hat{\mathbf{x}}$ are provided as input to the classifier. If the tokens in this case are generated using the *argmax* then this function is not differentiable. Hence, passing tokens effectively to the classifier is a challenge.

In (Yu et al., 2017), the REINFORCE (Williams, 1992) algorithm is used and rewards are calculated using Monte Carlo search sampling for the next tokens. This technique is known to be unstable due to the high variance of the sampled gradient during training (Shen et al., 2017). Kusner and Hernández-Lobato (2016) introduce the Gumbel-softmax distribution as a solution. It approximates the multinomial distribution parameterized in terms of the softmax distribution. Here the predicted token is:

$$\hat{\mathbf{x}}_t = \mathtt{softmax}(1/\tau(\hat{\mathbf{o}}_t + \mathbf{g}_t)),$$

where $\hat{\mathbf{o}}_t$ is described in (§6), τ is temperature parameter and $\mathbf{g}_t$ is sampled independently from the Gumbel distribution. Hu et al. (2017) use this technique without sampling from the Gumbel distribution but by only training the temperature parameter.

Combined module architecture: It is also possible to combine techniques from multiple modules to control the generation process. We mention some of the prior works that have successfully combined various modules here. Hu et al. (2017) combine stochastic changes (§2.2), KL Divergence loss (§6.2) and a classifier loss (§6.3). It adopts a variational auto-encoder along with KL divergence loss objective and further adds a discriminator loss which signifies if the generated sentence belong to the target attribute. As mentioned earlier, Romanov et al. (2019) combine the decomposition of the external input (§2.3) with external feedback provided to the external input (§2.4). External feedback is used to ensure that the decomposed latent sub-spaces represent the desired target attributes. Hu et al. (2018) establishes formal connections between generative adversarial networks (related to §5.2 and §6.3) and variational auto-encoders (related to §2.2 and §6.2). Determining the best possible combination of modules through empirical evaluation remains an open challenge.

8 Conclusion and Future Work

In this paper we propose a new schema to organize the prior work in controllable text generation. The schema contains five modules, each of which plays an important role in the generation process. We detail the various techniques used to modulate each of the five modules to perform controllable text generation. We also provide theoretical understanding and qualitative analysis of these techniques. This understanding paves way to new architectures based on combinations of these modules. The future work will focus on empirical comparison of these techniques to gain an insight into their usefulness and strength.

Acknowledgements

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

Vidhisha Balachandran, Artidoro Pagnoni, Jay Yoon Lee, Dheeraj Rajagopal, Jaime Carbonell, and Yulia Tsvetkov. 2020. Structsum: Incorporating latent and explicit sentence dependencies for single document summarization. *ArXiv e-prints*, 03.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, Berlin, Germany, August. Association for Computational Linguistics.

Khyathi Chandu, Shrimai Prabhumoye, Ruslan Salakhutdinov, and Alan W Black. 2019. "my way of telling a story": Persona based grounded story generation. In *Proceedings of the Second Workshop on Storytelling*, pages 11–21.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2019. Plug and play language models: a simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*.

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

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2018. Wizard of wikipedia: Knowledge-powered conversational agents. In *International Conference on Learning Representations*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia, July.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Chuang Gan, Zhe Gan, Xiaodong He, Jianfeng Gao, and Li Deng. 2017. Stylenet: Generating attractive visual captions with styles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3137–3146.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2018. A knowledge-grounded neural conversation model. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Hongyu Gong, Suma Bhat, Lingfei Wu, JinJun Xiong, and Wen-mei Hwu. 2019. Reinforcement learning based text style transfer without parallel training corpus. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3168–3180, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Junliang Guo, Xu Tan, Di He, Tao Qin, Linli Xu, and Tie-Yan Liu. 2019. Non-autoregressive neural machine translation with enhanced decoder input. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3723–3730.

Vrindavan Harrison, Lena Reed, Shereen Oraby, and Marilyn Walker. 2019. Maximizing stylistic control and semantic accuracy in nlg: Personality variation and discourse contrast. In *Proceedings of the 1st Workshop on Discourse Structure in Neural NLG*, pages 1–12.

Cong Duy Vu Hoang, Trevor Cohn, and Gholamreza Haffari. 2016. Incorporating side information into recurrent neural network language models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1250–1255.

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

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. Learning to write with cooperative discriminators. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1638–1649.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *Proc. ICML*, pages 1587–1596.

Zhiting Hu, Zichao Yang, Ruslan Salakhutdinov, and Eric P. Xing. 2018. On unifying deep generative models. In *International Conference on Learning Representations*.

Junjie Hu, Yu Cheng, Zhe Gan, Jingjing Liu, Jianfeng Gao, and Graham Neubig. 2019a. What makes a good story? designing composite rewards for visual storytelling. *arXiv preprint arXiv:1909.05316*.

Zhiting Hu, Haoran Shi, Bowen Tan, Wentao Wang, Zichao Yang, Tiancheng Zhao, Junxian He, Lianhui Qin, Di Wang, Xuezhe Ma, et al. 2019b. Texar: A modularized, versatile, and extensible toolkit for text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 159–164.

Qiuyuan Huang, Zhe Gan, Asli Celikyilmaz, Dapeng Wu, Jianfeng Wang, and Xiaodong He. 2019. Hierarchically structured reinforcement learning for topically coherent visual story generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8465–8472.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2019. Disentangled representation learning for non-parallel text style transfer. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 424–434, Florence, Italy, July. Association for Computational Linguistics.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 329–339, Austin, Texas, November. Association for Computational Linguistics.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *Proc. ICLR*.

Ilia Kulikov, Alexander Miller, Kyunghyun Cho, and Jason Weston. 2019. Importance of search and evaluation strategies in neural dialogue modeling. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 76–87, Tokyo, Japan, October–November. Association for Computational Linguistics.

Matt J Kusner and José Miguel Hernández-Lobato. 2016. Gans for sequences of discrete elements with the gumbel-softmax distribution. *arXiv preprint arXiv:1611.04051*.

Alex M Lamb, Anirudh Goyal Alias Parth Goyal, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In *Advances in neural information processing systems*, pages 4601–4609.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California, June. Association for Computational Linguistics.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. 2016b. A persona-based neural conversation model. In *Proc. ACL*.

Wei Li, Xinyan Xiao, Yajuan Lyu, and Yuanzhuo Wang. 2018. Improving neural abstractive document summarization with structural regularization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4078–4087, Brussels, Belgium, October-November. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2018. Learning structured text representations. *Transactions of the Association for Computational Linguistics*, 6:63–75.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. In *International Conference on Learning Representations*.

Lajanugen Logeswaran, Honglak Lee, and Samy Bengio. 2018. Content preserving text generation with attribute controls. In *Advances in Neural Information Processing Systems*, pages 5103–5113.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Aman Madaan, Amrith Setlur, Tanmay Parekh, Barnabas Poczos, Graham Neubig, Yiming Yang, Ruslan Salakhut-dinov, Alan W Black, and Shrimai Prabhumoye. 2020. Politeness transfer: A tag and generate approach. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1869–1881, Online, July. Association for Computational Linguistics.

Tong Niu and Mohit Bansal. 2018. Polite dialogue generation without parallel data. *Transactions of the Association for Computational Linguistics*, 6:373–389.

Thanapon Noraset, Chen Liang, Larry Birnbaum, and Doug Downey. 2017. Definition modeling: Learning to define word embeddings in natural language. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Nanyun Peng, Marjan Ghazvininejad, Jonathan May, and Kevin Knight. 2018. Towards controllable story generation. In *Proceedings of the First Workshop on Storytelling*, pages 43–49.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style transfer through back-translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 866–876, Melbourne, Australia, July.

Shrimai Prabhumoye, Chris Quirk, and Michel Galley. 2019. Towards content transfer through grounded text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2622–2632, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Shrimai Prabhumoye, Margaret Li, Jack Urbanek, Emily Dinan, Douwe Kiela, Jason Weston, and Arthur Szlam. 2020. I love your chain mail! making knights smile in a fantasy game world: Open-domain goal-oriented dialogue agents. *arXiv preprint arXiv:2002.02878*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Sudha Rao and Joel Tetreault. 2018. Dear sir or madam, may I introduce the GYAFC dataset: Corpus, benchmarks and metrics for formality style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 129–140.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press, USA.

Alexey Romanov, Anna Rumshisky, Anna Rogers, and David Donahue. 2019. Adversarial decomposition of text representation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 815–825, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proc. ACL*.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in neural information processing systems*, pages 6830–6841.

Akhilesh Sudhakar, Bhargav Upadhyay, and Arjun Maheswaran. 2019. "transforming" delete, retrieve, generate approach for controlled text style transfer. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3260–3270.

Jianheng Tang, Tiancheng Zhao, Chenyan Xiong, Xiaodan Liang, Eric Xing, and Zhiting Hu. 2019. Target-guided open-domain conversation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5624–5634.

Jack Urbanek, Angela Fan, Siddharth Karamcheti, Saachi Jain, Samuel Humeau, Emily Dinan, Tim Rocktäschel, Douwe Kiela, Arthur Szlam, and Jason Weston. 2019. Learning to speak and act in a fantasy text adventure game. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 673–683.

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

Ashwin K Vijayakumar, Michael Cogswell, Ramprasaath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2018. Diverse beam search for improved description of complex scenes. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Ke Wang, Hang Hua, and Xiaojun Wan. 2019a. Controllable unsupervised text attribute transfer via editing entangled latent representation. In *Advances in Neural Information Processing Systems*, pages 11034–11044.

Wenlin Wang, Zhe Gan, Hongteng Xu, Ruiyi Zhang, Guoyin Wang, Dinghan Shen, Changyou Chen, and Lawrence Carin. 2019b. Topic-guided variational auto-encoder for text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 166–177, Minneapolis, Minnesota, June. Association for Computational Linguistics.

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

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2020. Neural text generation with unlikelihood training. In *International Conference on Learning Representations*.

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

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Ziang Xie. 2017. Neural text generation: A practical guide. *arXiv preprint arXiv:1711.09534*.

Peng Xu, Yanshuai Cao, and Jackie Chi Kit Cheung. 2019. Unsupervised controllable text generation with global variation discovery and disentanglement. *CoRR*, abs/1905.11975.

Zichao Yang, Zhiting Hu, Chris Dyer, Eric P Xing, and Taylor Berg-Kirkpatrick. 2018. Unsupervised text style transfer using language models as discriminators. In *Advances in Neural Information Processing Systems*, pages 7287–7298.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764.

Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7378–7385.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-first AAAI conference on artificial intelligence*.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213. Association for Computational Linguistics.

Kangyan Zhou, Shrimai Prabhumoye, and Alan W Black. 2018. A dataset for document grounded conversations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 708–713.

Yiheng Zhou, Yulia Tsvetkov, Alan W Black, and Zhou Yu. 2020. Augmenting non-collaborative dialog systems with explicit semantic and strategic dialog history. In *International Conference on Learning Representations*.