Final Review Project
Understanding Controllable Text Generation

Yutong Shen
ID: 2020403405

***Abstract*—Controllable text generation, under the research topic of natural language generation in the field of natural language processing, is gaining attentions due to its broad application. Developing from the typical text generation task that produces understandable texts in human languages based on data, controllable text generation further aims to generate texts whose attributes can be controlled. Current researches present varying methods for controllable text generation. Most of the researches are task-specific, without an approach-based framework as a classification for the technique used. This review project aims to explore a way to classify different research techniques into six main modules to form a systematic understanding towards the topic of controllable text generation. I present the overview of techniques with a summary of research goal of each paper. The detailed explanation and evaluation are considered, as well as an analysis for the open problem and future work.**

I. Introduction

Controllable text generation, under the research topic of natural language generation in the field of natural language processing, is gaining attentions due to its broad application. Developing from the typical text generation task that produces understandable texts in human languages based on data, controllable text generation further aims to generate texts whose attributes can be controlled.

To illustrate controllable text generation mathematically, the task could be viewed as a language modeling problem:

$$P(Y \mid X) = \prod_t P(y_t \mid y_1, y_2, ..., y_{t-1}, X)$$

where X is the input or source sequence and Y is the output to be generated or the target sequence. The probability distribution of Y is equivalent to a conditional probability. At each time step t, the model takes in current input x_t and previous hidden state to generate output y_t and make prediction.

The attributes being controlled can range from content(keyword, information, etc.), style(sentiment, politeness, etc.), or personal information(gender, personality,etc)(Prabhumoye et al. [26]) . The variety of attributes leads to diverse ways of applying controllable text generation. For instance, Peng et al. [23] applied controllable text generation to write stories based on happy/sad ending choices;Wang et al. [38] collected social media comments and writers' personalities data for rewriting comments with given personalities; Zhou et al. [48] tended to form the negotiation text with specific strategy for simulating actual human response. In general, the application of controllable text generation is similar to those of typical text generation, including Machine translation, Dialogue Systems, Story Telling, Poetry Generation, and Text Summarization, but with the diversity of controlled attributes.

Due to its wide application, a large amount of research papers have been done for controllable text generation, with combinations of different controlled attributes and types of task. Most of the research papers for controllable text generation are task specific, which means overlap could exist between the approaches they introduced, and similar approaches could be applied to different tasks. Thus, a paper review could be done to categorize these papers and evaluate techniques. There is very limited number of review paper compared to the capacity of research papers for controllable text generation. Among the review paper, Prabhumoye et al. [26] introduced a unifying schema, including five modules of external input, sequential input, generator operations, output, and training objective along the generation process. The paper provides an insight to the current developments and contributions for controllable text generation, but without detailed explanation for the examples given. Weng [40] provided another way of classification in the online blog post, but the classification is based on the assumption that a pre-trained language model already exists, so actually many methods are not considered, such as the types of pre-trained language model. Therefore, I have had a motivation of writing a review paper based on the paper I have read related to controllable text generation.

For this review paper, I got inspirations from the structure professor Liu had in class for the topic of text generation. He introduced the topic based on the type of language models and their applications. Therefore, I tended to follow the logic to make classification, as well as modifying with more approaches following the generation process to propose a review for the research area of controllable text generation. The main modules for this paper is approach-based, not task-based, including Variational Auto-Encoder(VAE), Recurrent Neural Network(RNN), Pre-trained Language Model(PLM), Attention and Transformer, Fine-tuning, and Training Objectives. Variational Auto-Encoder(VAE) is used for updating the initialization of the generator; Recurrent Neural Network(RNN) and Pre-trained Language Model(PLM) could take the input and return an output through generative operations; Attention and Transformers mainly focus on the output and help make predictions; Fine-tuning helps to optimize the output by guiding a language model; Training objectives control the generation with objective functions.

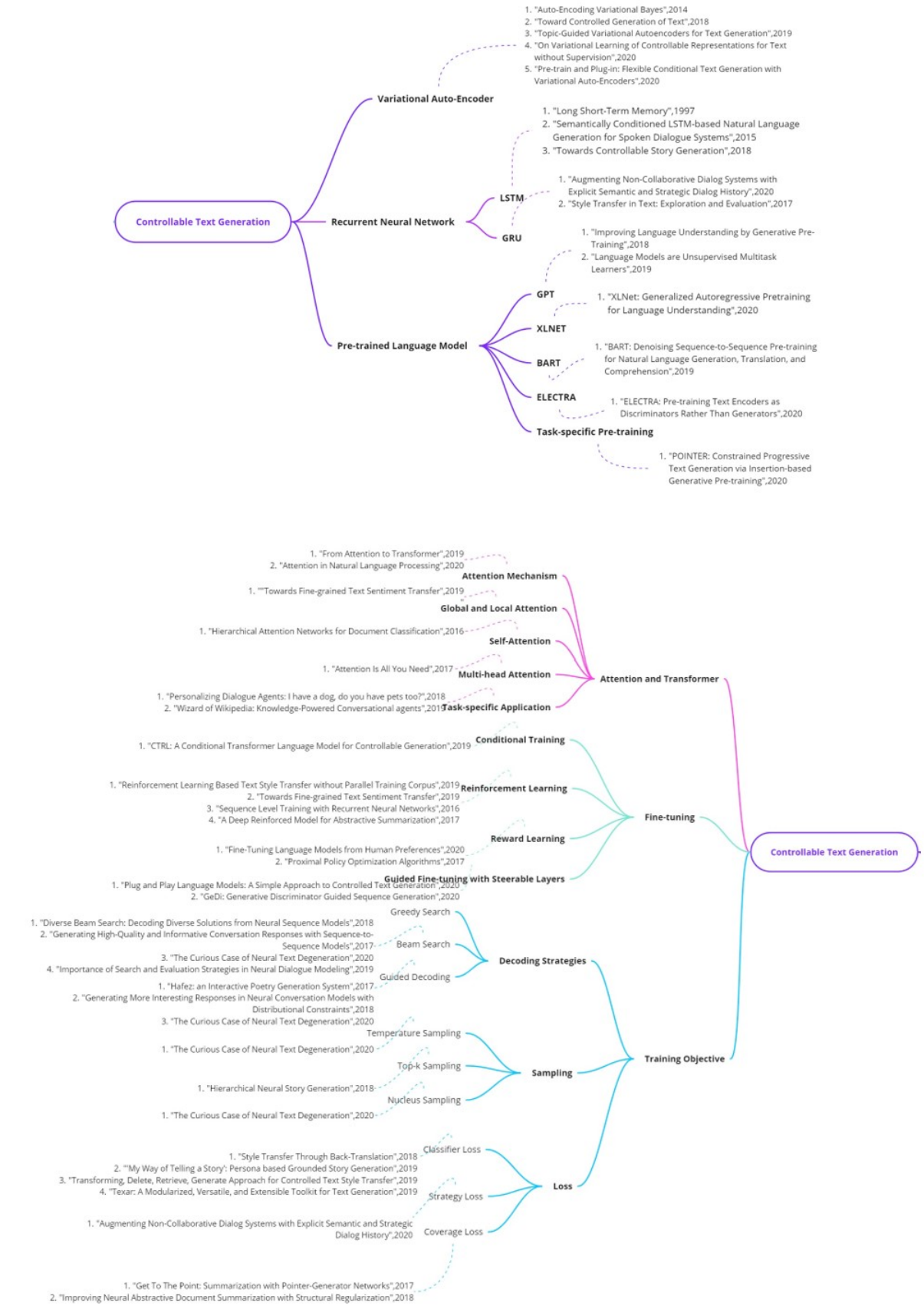

Fig. 1. Schema of Review Content for Controllable Text Generation

II. Related Work

Controllable text generation is based on the research area of text generation, with modification of adding controlled attributes. Text generation aims to generate human understandable text based on data. As taught in course, application for text generation can be categorized into few types: Machine translation, Dialogue Systems, Story Telling, Poetry Generation, and Text Summarization. Text generation could be further viewed as traditional text generation and neural text generation. Rule-based and statistical models for traditional text generation have advantages of being fairly interpretable and well-behaved but need hand-engineering. Neural text generation requires less hand-engineering but has higher performance. Similarly, the trade-off could be applied to controllable text generation, and normally a hymbrid way is applied for real generation tasks. The content of this review project will mainly focus on controllable neural text generation but also includes some human engineering methods such as asking human to add preference label through data analysis(Ziegler et al. [49]). In addition, the approaches for neural text generation could be divided as autoregressive and non-autoregressive, which could be applied for controllable text generation. In addition to the autoregressive methods such as language modelling, this review paper also includes non-autoregressive method such as POINTER(Zhang et al. [47]). In general, the topic of controllable text generation is closely related to text generation but with more constraints.

III. Review Content

As shown in figure1, I split the topic of controllable text gneration into six main methods, including Variational Auto-Encoder(VAE), Recurrent Neural Network(RNN), Pre-trained Language Model(PLM), Attention and Transformer, Fine-tuning, and Training Objectives. The following content would provide further illustration for each module.

A. Variational Auto-Encoder(VAE)

Firstly introduced by Kingma and Welling [16], Variational Auto-Encoder(VAE) is a powerful generative model for generation tasks. It could be viewed as a regularized version of standard auto-encoder that introduces latent variables to auto-encoder architecture. Instead of learning a single point from data, VAE learns a region over latent space. A distribution $q(z \mid x)$ is encoded. KL divergence is used to make the $q(z \mid x)$ close to the prior $p(z)$. The motivation is that if only reconstruction loss is used, the variances of $q(z \mid x)$ will still be small (almost the same as the original single point).

Hu et al. [13] aimed at generating plausible text sentences, whose attributes are controlled by learning disentangled latent representations with designated semantics. The author proposed a new neural generative model which combines VAEs and holistic attribute discriminators for effective imposition of semantic structures. The model can alternatively be seen as enhancing VAEs with the wake-sleep algorithm for leveraging fake samples as extra training data. The wake-sleep algorithm is similar to extended sleep procedure of inforGan.

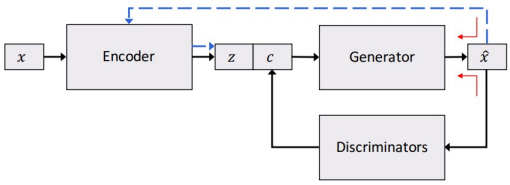

Fig. 2. Model OverviewVAE+(generator+discriminator).(Image source:[13])

For each attribute code, an individual discriminator is set up to measure how well the generated samples match the desired attributes, and drive the generator to produce improved results. The difficulty of applying discriminators in context is that text samples are discrete and non-differentiable, which breaks down gradient propagation from the discriminators to the generator. A continuous approximation based on softmax is used with a decreasing temperature, which anneals to the discrete case as training proceeds. This simple yet effective approach enjoys low variance and fast convergence. The limitation for this model is that the sentence length is too short and there is no provided solution for combining attributes together. More experiments could be done to apply the model to other NLP tasks such as dialogue generation.

Wang et al. [37] proposed a topic-guided variational autoencoder (TGVAE) model for text generation. Distinct from existing variational autoencoder (VAE) based approaches, which assume a simple Gaussian prior for the latent code, the model specifies the prior as a Gaussian mixture model (GMM) parametrized by a neural topic module. Each mixture component corresponds to a latent topic, which provides guidance to generate sentences under the topic. The model mainly consists two modules: the Neural topic model(NTM) for capturing long-range semantic meaning across the document, and the Neural Sequence model(NSM) for generating a sentence with designated topic guidance.

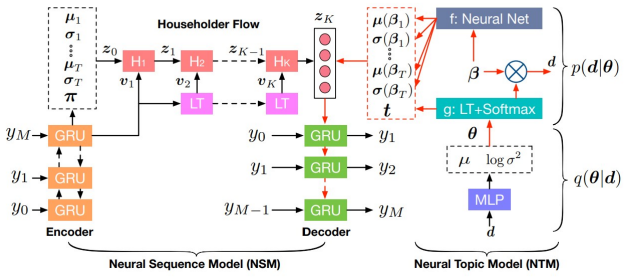

Fig. 3. The model schema.(Image source:[37])

Xu et al. [42] proposed Constrained Posterior VAE(CP-VAE), which constrain the posterior mean to a learned probability simplex and only perform manipulation within the probability simplex, to mitigate the latent vacancy problem on text(vacant regions in the latent code space not being considered by the decoding network). For CP-VAE, VAE is used

for sentiment attribute manipulation. Duan et al. [5] presented Pre-train and Plug-in Variational Auto-Encoder (PPVAE), a framework for flexible conditional text generation based on VAE. PPVAE decouples the text generation module from the condition representation module to allow "one-to-many" conditional generation. When a fresh condition emerges, only a lightweight network needs to be trained and works as a plug-in for PPVAE, which is efficient and desirable for real-world applications. The model consists of two modules, PRETRAINVAE and PLUGINVAE. PRETRAINVAE is used to encode and generate text, while PLUGINVAE is used to learn the transformation between the conditional and global latent space for each condition. The condition ranges from sentiment, length, and topic.

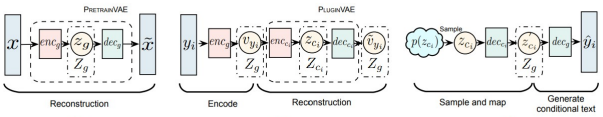

Fig. 4. Workflow for PPVAE framework.(Image source:[5])

Although VAE works effectively in the field of controllable text generation when the control attributes can be represented as latent variables such as style, topic and sentiment, it still has limitation for content grounded text generation tasks where specific information, keywords or entities have to guide the generation.(Prabhumoye et al. [26])

B. Recurrent Neural Network(RNN)

Recurrent Neural Network (RNN) is a class of neural network that utilizes the sequential data. As per taught in class by professor Liu, RNN introduces the concept of sequential memory, which enables it to remember past information, influencing its decisions at current step. Multiple cells within RNN share the same parameter, so theoretically RNN could process sequences with arbitrary length without increasing its model length. However, the number of steps RNN is able to look back are limited empirically. In order to address this issue, Long Short-Term Memory Network(LSTM) and Gated Recurrent Unit (GRU) are developed as a improvement over RNN. Therefore, RNN still provides an effective model for solving NLP problems. In fact, many researches have based on RNN model(LSTM or GRU) in the field of Controllable Text Generation.

1) Long Short-Term Memory Network(LSTM):
Long Short-Term Memory (LSTM) is a special type of RNN that has wide applications(Hochreiter and Schmidhuber [11]). Apart from the original RNN, LSTM introduces an additional hidden state (cell state) to solve the vanishing gradient issue. The gate mechanism on cell states allows information to flow optionally by letting the 'forget gate layer' and 'input gate layer' to decide which information to forget and which to keep. Three of these gates is contained in LSTM for maintenance. The cell state is then updated and the information is filtered for output.

Wen et al. [39] presented a statistical language generator based on a semantically controlled Long Short-term Memory (LSTM) structure. The core of the paper is that it improves

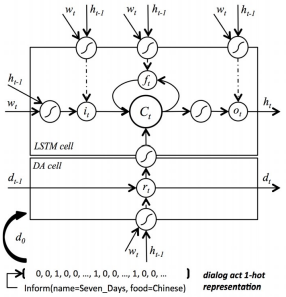

Fig. 5. Semantic Controlled LSTM cell.(Image source:[39])

LSTM by involving DA(dialogue act) system for key-words controll. Key-words are encoded using one-hot representation. An additional control cell is introduced into the LSTM to gate the DA. This cell plays the role of sentence planning since it manipulates the DA features during the generation process in order to produce a surface realisation which accurately encodes the input information. At each time step the DA cell decides what information should be retained for future time steps and discards the others.

$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t + tanh(W_{dc} d_t)$$

where c_t is the cell value that depends on DA, and d_t is the one-hot representation of DA.

Peng et al. [23] presents a general framework of analyzing existing story corpora to generate controllable and creative new stories. The framework is applied to build recurrent neural network (RNN)-based generation models to control story ending valence.

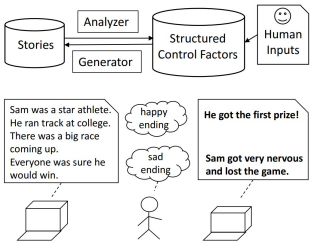

Fig. 6. An overview (upper) and an example (lower) of the proposed story framework.(Image source:[23])

The feature for classifier is a LSTM-cell, and the forget gate and input gate of LSTM decides whether the information from input story to be kept or discarded. The generator for story line controlled generation is also LSTM based, where the story line words are encoded into vectors by BiLSTM and decoded by the LSTM-cell with an attention layer.

2) Gated Recurrent Unit (GRU):
Gated Recurrent Units (GRU) is another improved version of RNN with gate mechanism. Compared with LSTM, GRU solves vanishing gradient issue with a different but simplified structure. The update gate enables GRU to remember important information and the reset gate enables GRU to forget unimportant information.

Zhou et al. [48] studied non-collaborative dialogs and built a model called finite state transducers (FSTs) for dialogue generation(negotiation task) with target strategy. A standard GRU is used to encode current utterance into a hidden state, while another to combine all utterances till current time to encode the entire dialog into a hidden state. A standard GRU with attention(utterance decoder) is also used for decoding process. The utterance decoder, combined with strategy predictor, is able to guide dialogue generation with specific strategy.

For style transfer task, Fu et al. [7] also used GRU for both encoding and decoding process. The model proposed tends to solve the lack of parallel data for style transfer.

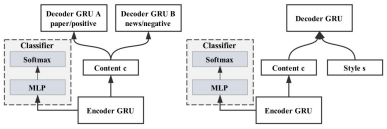

Fig. 7. Two models proposed with GRU.(Image source:[7])

C. Pre-trained Language Model(PLM)

Currently many pre-trained language models are used for generating higher quality text with controlled attributes, this section would present the frequent used pre-trained language models.

1) GPT:
GPT has gained attention as a powerful model. Radford and Narasimhan [27] presented that the training procedure of GPT consists two stages. The first stage is performing unsupervised pretraining a high-capacity language model on a large corpus of text. Transformer decoder is used within the language model. The second stage is performing supervised fine-tuning, where we adapt the model to a discriminative task with labeled data.

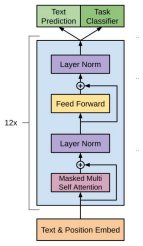

Fig. 8. Transformer architecture and training objectives used in this work.(Image source:[27])

Radford et al. [28] continued to make improvement from GPT. GPT-2 largely follows the details of the GPT model, but with a few modifications. Layer normalization is moved to the input of each sub-block, similar to a pre-activation residual network and an additional layer normalization was added after the final self-attention block. A modified initialization which accounts for the accumulation on the residual path with model depth is used. The datasets size for GPT-2 increases with a expanded vocabulary and context size.

GPT demonstrates its significant improvement and has a wide application for various NLP tasks, including text generation.

2) XLNET:
Yang et al. [44] proposed XLNET, which is a generalized autoregressive pretraining method (1) enables learning bidirectional contexts by maximizing the expected likelihood over all permutations of the factorization order and (2) overcomes the limitations of BERT thanks to its autoregressive formulation. Furthermore, XLNet integrates ideas from Transformer-XL, the state-of-the-art autoregressive model, into pretraining. Experiments have shown than the computation is very expensive for XLNET, which could be reduced by using pre-trained model and fine-tuning.

3) BART:
Lewis et al. [19] proposed BART, a denoising auto-encoder for pre-training sequence-to-sequence models. Compared with GPT that uses the decoder part of transformer, BART uses the encoder part. Pre-training of BART consists two stages: an arbitrary noising function that corrupts text and a sequence-to-sequence model that reconstructs the original text. BART uses a standard Tranformer-based neural machine translation architecture which, despite its simplicity, can be seen as generalizing BERT (due to the bidirectional encoder), GPT (with the left-to-right decoder), and many other more recent pretraining schemes. BART is particularly effective when fine tuned for text generation.

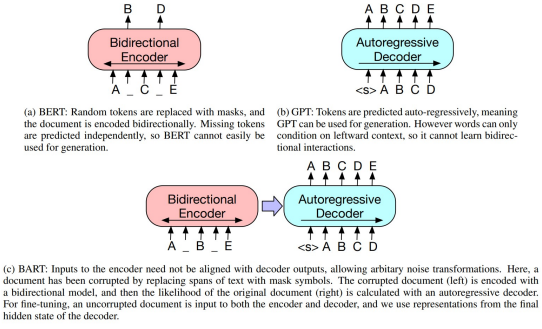

Fig. 9. A schematic comparison of BART with BERT and GPT.(Image source:[19])

4) ELECTRA:
Clark et al. [2] thought that MLM only learns from 15

$$p_G(x_t \mid x) = exp(e(x_t)^T h_G(x)_t) / \sum_{x'} exp(e(x')^T h_G(x)_t)$$

Then the discriminator judges whether all words are replaced:

$$D(x,t) = sigmoid(w^T h_D(x)_t)$$

Although similar to the training objective of a GAN, the

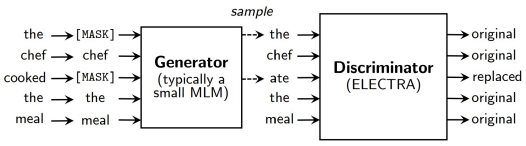

Fig. 10. : An overview of replaced token detection.(Image source:[2])

major difference between ELECTRA and GAN is that the generator is trained with maximum likelihood rather than being trained adversarially to fool the discriminator. Thus, compared to masked language modeling, the pre-training objective of ELECTRA is more compute-efficient and results in better performance on downstream tasks.

5) Task-specific Pre-training:

Zhang et al. [47] introduced POINTER as an insertion-based non-autoregressive model for hard-constrained text generation. POINTER allows long-term control over generation and reduces the empirical time complexity from $O(n)$ to $O(logn)$ at best. Large-scale pre-training and novel beam search algorithms are proposed to further boost performance. POINTER utilizes a DP-base approach for data pair construction by discarding words from token:

$$\max \sum_{t=1}^{T} -\phi_t \cdot \alpha_t \quad s.t. \forall t < T(1-\phi_t)(1-\phi_{t+1}) \neq 1$$

Starting from the lexical constraints (X^0), at each stage, the algorithm inserts tokens progressively to formulate the target sequence. At each step, at most one new token can be generated between two existing tokens. Factorize the distribution according to the importance of each token and the final token at step k is $X = X^K$:

$$p(X) = p(X_0) \prod_{k=1}^{K} p(X^k | X^{k-1})$$

where $p(X^k|X^{k-1}) = \prod_{x \in X^k - X^{k-1}} p(x|X^{k-1})$.

There are many research papers with pre-trained model and fintuning methods for controllable text generation. For instance, Urbanek et al. [34] introduced a large-scale crowdsourced fantasy text adventure game research platform where agents—both models and humans—can act and speak in a rich and diverse environment of locations, objects, and other characters, Wu et al. [41] proposed a framework called controllable grounded response generation (CGRG), in which lexical control phrases are either provided by an user or automatically extracted by a content planner from dialogue context and grounding knowledge for performing flexible semantic control in response generation process.

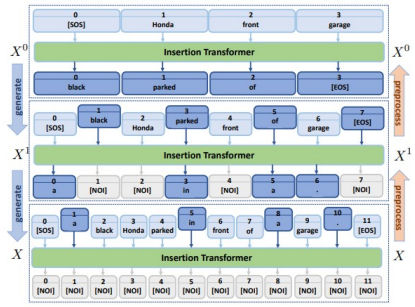

Fig. 11. Illustration of generation process of POINTER.(Image source:[47])

Section E includes more fine-tuning methods with pre-trained model, such as CTRL and PPLM.

D. Attention and Transformer

Attention mechanism has been widely applied to NLP tasks. It is a part of a neural architecture that enables to dynamically highlight relevant features of the input data, typically a sequence of textual elements.(Galassi et al. [8]) Attention could help improve the performance of Seq2seq model, enabling the decoder to directly look at the source sentence. Attention also provides shortcuts to long-distance states and solve vanishing gradient problem in training RNNs. In addition, transformer is based on attention mechanism to guide generation. It is able to parallel compute and gain the effectiveness of attention mechanism.

1) Attention Mechanism:

The attention mechanism utilizes each word $x_1, x_2, ..., x_n$ as input token. The input tokens are fed to RNN to get hidden states $h_1, h_2, ..., h_n$. By generating context vector $c_1, c_2, ..., c_n$ at each time step from each corresponding hidden state, it is ensure that the length of context vector used by decoder is varied. Improving from the probability distribution of traditional seq2seq model:

$$p(y_t \mid y_1, ..., y_t - 1, c) = g(y_{t-1}, s_t, c)$$

The probability distribution of attention:

$$p(y_i \mid y_1, ..., y_i - 1, x) = g(y_{i-1}, s_i, c_i)$$

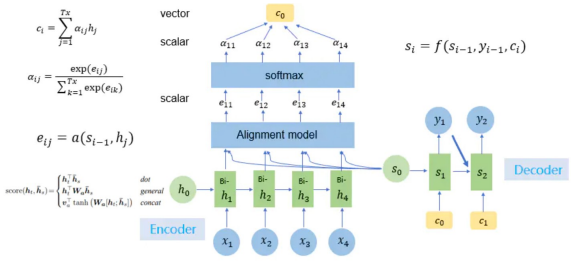

Fig. 12. Attention Mechanism.(Image source:[43])

2) Global Attention and Local Attention:
Luo et al. [21] proposed *global attention* and *local attention* to improve the generation of context vector c. Similar to attention, but using stacking LSTM. To further increase the efficiency of calculation, *local attention* only focus on words close to the target word. For every target word, *local attention* generates a corresponding position p_t and gets the context vector from hidden layer in the range $[p_t - D, p_t + D]$, where D is the window size.

3) Self-Attention:
Yang et al. [45] introduced self-attention that takes in parameter W for calculating scores, but not the decoder as attention does. For every hidden layer h, there is a corresponding score. Self-attention is generally used for the text classification task that eable model to find key words that would effect the classification result.

4) Multi-head Attention:
Vaswani et al. [35] defined attention as a mapping from a query and a set of key-value pairs to an output. The query, key, value and corresponding output are all vectors. The input and output of attention do not rely on RNN anymore. Thus the paper provides a clear understanding for the nature of attention. Two types of attention are defined in the paper: *scaled dot-product attention* and *multi-head attention*.

For *scaled dot-product attention*, the input consists of queries and keys of dimension d_k, and values of dimension d_v. The dot product of the query is computed with all keys, divide each by $\sqrt{d_k}$, and a softmax function is applied to obtain the weights on the values:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

In practical cases, the output metric, is computed on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V.

Multi-head attention is the addition of multiple *scaled dot-product attention*. Adding h times is equivalent to apply linear transformation on (Q, K, V) to obtain h numbers of (Q_i, K_i, V_i). The attention function is applied to the (Q_i, K_i, V_i) in parallel and the outputs are concatenated and applied linear transformation to get the final values:

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O$$
$$where head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

where the projections(linear transformation) are parameter matrices W_i.

5) Task-specific Application:
Zhang et al. [46] presented the task of making chit-chat more engaging by conditioning on profile information. Based on the given profile information and the persona information, the model could make utterance prediction. Then attention mechanism is applied to the ranking profile memory network

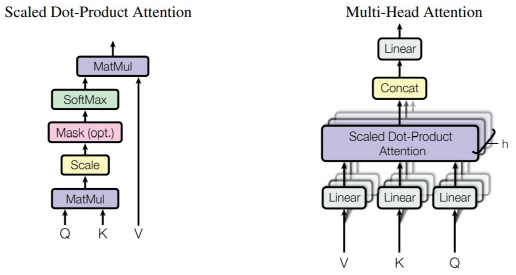

Fig. 13. (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.(Image source:[35])

for adding profile information and key-value profile memory network to dialog generation with persona. However, the model does not consider the chit-chat setting and is more concerned with achieving functional goals than displaying a personality. Many of the tasks and datasets available are constrained to narrow domains.

Dinan et al. [4] considered the task of open-domain dialog generation. For the model architecture, the author utilizes two baseline models, the retrieval transformer memory network and generative transformer memory network to guide dialogue response generation. Dialogue context and knowledge are encoded using a shared encoder from transformers. An attention mechanism to perform fine-grained selection of which knowledge sentences will be used to produce the next turn of dialogue.

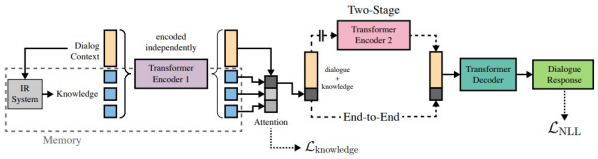

Fig. 14. Generative transformer memory network.(Image source:[4])

Recently, the domains for development of using attention weights to control over generation still exists for attributes like style, topic, content,etc.(Prabhumoye et al. [26])

E. Fine-Tuning

Based on Section C, this section presents more fine-tuning method with pre-trained models to guide text generation process with attributes control.

1) Conditional Training:
CTRL is a 1.63 billion-parameter conditional transformer language model, trained to condition on control codes(Keskar et al. [15]). CTRL learns the conditioned distribution $p(x \mid a)$ by training on raw text sequences with control code prefixes, such as [horror], [legal], etc. Then the learned model is able to generate text with respect to the prompt prefix. The key of CTRL is that it uses raw text from various sources such as WikiPedia, where each dataset is assigned with a control

code, which enhances its pertinence while controlling over the generating process. Based on transformer, CTRL has a similar structure, but including the control code c to learn the distribution $p(x \mid c)$:

$$\prod_{i=1}^{n} p\left(x_i \mid x_{<t_i}, c\right)$$

Starting with the same prompt, different control code could lead to related results separately. Through control code, one could more explicitly influence generation with style, genre, content, and other task-specific domain.

Horror *A knife* handle pulled through the open hole in the front. I jumped when the knife hit.\n\nEyes widened in horror. Her scream was the only sound I heard besides her sobs.\n\nThe spider touched her feet as it started to dig into the top of her arch. The creature's tears began to flow.\n\nThe spider looked up to her and looked back at me with eyes filled with burning tears. My heart started to race. . .

Reviews *A knife* is a tool and this one does the job well.\n\nRating: 4.0\n\nI bought these for my husband who has been using them to cut up his own meat since he got them. He says they are very sharp so be careful when you use them, but that doesn't seem like much of an issue because he's used it on everything from chicken breasts to beef tenderloin. . .

Fig. 15. Conditional generation sample by CTRL.(Image source:[15])

However, CTRL doesn't consider what should not be generated, and many domains are mapped into one single control code. For the space of improvement, it could be refined to let the model have more fine-grained control.

2) Reinforcement Learning:
Reinforcement learning has recently been applied to NLP tasks. In the field of style transfer, Gong et al. [10] proposed a reinforcement-learning-based text style transfer system without accessing parallel corpus. The main model is based on Encoder-Decoder. By considering style, semantic, and fluency scores as reward as rewards, the reinforcement learning method is used to optimize the Encoder-Decoder model.

Luo et al. [21] focused on the task of fine-grained control of sentiment for controllable text generation. A cycle reinforcement learning algorithm is used to tackle the lack of parallel data for guiding generation. A reward function is designed for transforming the sentiment of text while preserving content. In addition, based on the idea of back-translation, it is ensured that the sentences generated in the forward direction can also be generated in the source style sentence through the reverse operation.

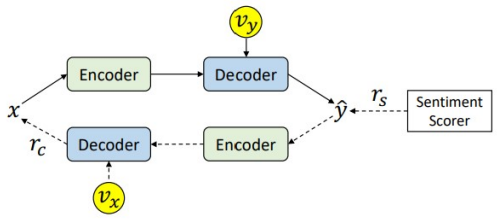

Fig. 16. Cycle Reinforcement Learning:(Image Source: [21])

Ranzato et al. [29] applied the Reinforcement-Learning algorithm to the problem of sequence generation. The model used for training is the traditional RNN. The author casted the sequence generation task in reinforcement learning framework, viewing the RNN model as agent, its parameters as a policy, and execution results in the agent picking as action. For training, the Cross-entropy (XENT) loss is computed for the first s steps, and reinforcement in the remaining steps.(The model is fine-tuned with Cross-entropy loss and reinforcement learning). The fine-tuning method based on reinforcement learning is a modification to address the deficiencies (exposure bias, loss does not operate at the sequence level) in training current generative model.

Similarly, Paulus et al. [22] combined the maximum-likelihood cross-entropy loss with rewards from policy gradient reinforcement learning to reduce exposure bias. The loss for maximum-likelihood training:

$$\mathcal{L}_{ml} = -\sum_{t=1}^{n'} \log p(y_t{}^* \mid y_1{}^*, ..., y_{t-1}{}^*, x)$$

Reinforcement learning is applied to maximize an evaluation metric, usually ROUGE-1, using self-critical policy gradient training. For the training algorithm, two separate output sequences are produced at each training iteration, y^s and $\hat{y}$. y^s is obtained by sampling from probability distribution at decoding time step, and $\hat{y}$ is obtained by maximizing output probability distribution at each time step(greedy search). The loss function for reinforcement learning:

$$\mathcal{L}_{rl} = (r(\hat{y}) - r(y^s)) \sum_{t=1}^{n'} \log p(y_t^s \mid y_1^s, ..., y_{t-1}^s, x)$$

where $r(y)$ is the reward function for output sequence y, comparing with y^* as the ground truth sequence with evaluation metric.

The training function is a fixture of two types of loss function:

$$\mathcal{L}_{mixed} = \gamma \mathcal{L}_{rl} + (1 - \gamma)\mathcal{L}_{ml}$$

γ is a scaling factor for the difference in magnitude between $\mathcal{L}_{rl}$ and $\mathcal{L}_{ml}$.

3) Reward Learning:
Fine-Tuning Language Models from Human Preferences advances generative pre-training of language models to apply reward learning to datasets(Ziegler et al. [49]). It collected human labels by asking humans to select the best candidate y_0 out of a few options y_i given the input $x \sim \mathcal{D}$, with select option b.

Having collected a dataset S, a reward model r is initialized and trained using the loss function:

$$loss(r) = \mathbb{E}_{(x,\{y_i\},b)\sim\mathcal{S}}\left[\log \frac{e^{r(x,y_b)}}{\sum_i e^{r(x,y_i)}}\right]$$

To keep the scale of the reward model consistent across training, it is normalized to have a mean 0 and variance 1 for $x \sim \mathcal{D}, y \sim \rho(.|x)$.

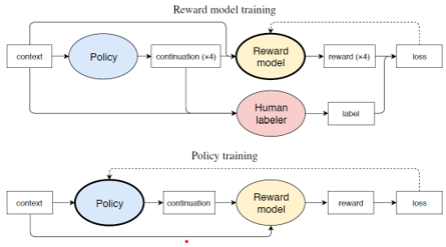

Fig. 17. Training process for reward model and policy.(Image source:[49])

The policy π, is trained via Proximal Policy Optimization(PPO, Schulman et al. [30]) with the above reward model during RL fine-tuning. The KL penalty to avoid policy's deviation:

$$R(x, y) = r(x, y) - \beta \log \frac{\pi(y|x)}{\rho(y|x)}$$

Additional samples are collected in the online data collection case, to retrain the reward model periodically.

Although achieving good results based on human evaluation, one major limitation human labeling might have very high disagreement when ground truth is fuzzy.

4) Guided Fine-tuning with Steerable Layers:
Instead of training on the entire LM like CTRL does, PPLM proposed a simpler approach(Dathathri et al. [3]). The idea is that the target model $p(x \mid a)$ is proportional to $p(a \mid x)p(x)$. PPLM combines one or multiple simple attribute models $p(a \mid x)$ with a pre-trained, unconditional language model $p(x)$. The process is in three phase: At Step 1, a forward pass is performed through the language model to compute the likelihood of a using $p(a|x)$. At Step 2, a backward pass updates the internal latent representations of the LM, using gradients from the attribute model, to increase the likelihood of the passage having the desired attribute. At Step 3, a new distribution over the vocabulary is generated from the updated latents and the current token. The next token is then sampled from the updated distribution.

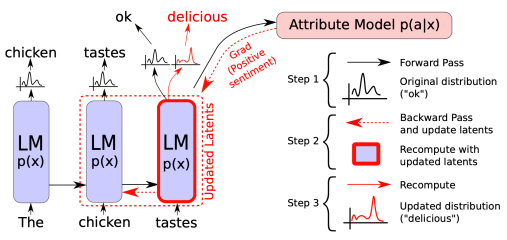

Fig. 18. Simplified illustration of the approach in three phases for PPLM.(Image source:[3])

Let H_t be the hidden state at step t, a is the desired attribute, x is the next token, then ΔH_t is the stepwise update to H_t, such that $(H_t + \Delta H_t)$ shifts the generated text closer to having a. ΔH_t is initialized at zero, and updates with gradients from the attribute model. Then, attribute model $p(a \mid x)$ could be rewrite as: $p(a \mid H_t + \Delta H_t)$:

$$\Delta H_t \leftarrow \Delta H_t + \alpha \frac{\nabla_{\Delta H_t} \log p(a|H_t + \Delta H_t)}{\|\nabla_{\Delta H_t} \log p(a|H_t + \Delta H_t)\|^{\gamma}}$$

Finally, it uses the updated $\tilde{H}_t = H_t + \Delta H_t$ to get $\tilde{o}_{t+1}, H_{t+1} = LM(x_t, \tilde{H}_t)$, for predicting the next token $x_{t+1} \sim \tilde{p}_{t+1} = softmax(W\tilde{o}_{t+1})$. This process of updating the latents is repeated at each time-step, leading to a gradual transition towards the desired attribute. The paper also proposed two types of attribute models: bag of words(BoW) and single layer discriminator.

Two methods are used to ensure the fluency of generated text: Minimize the Kullback–Leibler (KL) Divergence between H_t and $\tilde{H}_t$, times coefficient $\lambda_K L = 0.01$; Perform Post-norm Geometric Mean Fusion to approach $p(x)$. $x_{t+1} \sim 1/\beta(\tilde{p}_{t+1}^{\gamma_g m} p_{t+1}^{1-\gamma_g m})$. If $\gamma_g m \to 1$, the output converges to the LM after updates. If $\gamma_g m \to 0$, the output converges to the LM before updates. $\gamma_g m$ is set in $[0.8, 0.95]$.

The limitation of PPLM is that it might cause long test time due to multiple passes.

GeDi is also discriminator-guided, without requiring directly fine-tuning the LM(Krause et al. [17]). With Class-conditional language models(CC-LM), GeDi generalizes control of desired attributes with control code c, while guiding models away from undesired attributes with anti-control code $\bar{c}$. GeDi uses the contrast between $p_\theta(x_{1:t}|c)$ and $p_\theta(x_{1:t}|\bar{c})$ to compute the probability that every candidate next token x_t belongs to the desired class.

$$p_\theta(c|x_{1:t}) = \frac{p(c)p_\theta(x_{1:\tau}|c)^{\alpha/\tau}}{\sum_{c' \in \{c,\bar{c}\}} p(c')p_\theta(x_{1:\tau}|c')^{\alpha/\tau}}$$

where $p(c) = \exp(b_c)/\sum_{c'} \exp(b_{c'})$, and b_c is a prior learned class. GeDi-guided generation tends to have a high efficiency

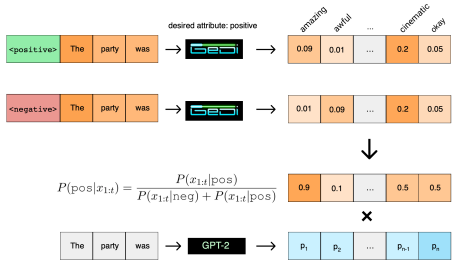

Fig. 19. Illustration of how GeDi-guided generation uses Bayes rule.(Image source:[17])

with Baye's Rule(30× faster than applying PPLM with GPT2-XL using default settings)

F. Training Objective

Training process is an important part for generating high-quality text, and by alternating different training strategies, such as various decoding, different sampling mehods, or changing the choice of loss function, would lead to better model performance.

1) Decoding Strategies:

For the Natural Language Generation(NLG) task, the model could be split into two parts: encoder and decoder. The encoder produces a representation of the source sentence and the decoder generates target sentence conditioned on encoding. Decoding strategies is an important component of the decoding process and text generation.

a) Greedy Search:

Based on the decoding process, $argmax_{y_i} P(y_i \mid y_1, ..., y_{i-1}, x)$, the probability of each word is computed at each step. Greedy Search simply chooses the word with the highest probability at each step. However, Greedy could not generate globally optimal solution, and its complexity is too high in practice.

b) Beam Search:

Beam Search is another widely used method for decoding. It stores the top-B highly scoring candidates at each time step, where B is known as the beam width. The top-B candidates will move to next time step and the process repeats, so the model will store B candidates and choose the most likely one at the end. However, Beam Search also could not generate globally optimal solution. Beam Search seems to produce text with 'endless' repetition, as mentioned by Vijayakumar et al. [36] and Shao et al. [32]. Holtzman et al. [12] compared the probability assigned to text generated by Beam Search and Humans, presenting the repetition issue of Beam Search.

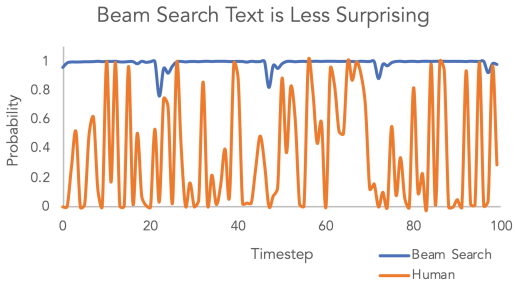

Fig. 20. Plot of Probability Comparison.(Image source:[12])

Beam Search variants have been proposed to optimize Beam Search. Diverse Beam Search divides the single set of beams into groups and encourage diversity only between groups and not within(Vijayakumar et al. [36]). It produces sequences with more variances. Iterative Beam Search could radically increase the size of search space without introducing much computational overhead(Kulikov et al. [18]). In their experiments, Iterative Beam Search could produce more diverse responses than Beam Search.

c) Guided Decoding:

Several developments have been made to the standard decoding methods to guide the generation. Ghazvininejad et al. [9] introduced an automatic poetry generation system 'Hafez' that enables users to revise and polish generated poems by adjusting various style configurations. To control the style, they modified the beam search process at decoding step as:

$$score(w, b) = score(b) + \log P_{RNN}(w) + \sum_i w_i * f_i(w);$$
$$\forall w \in V_{suc}$$

where $\log P_{RNN}(w)$ is the log-probability of word w calculated by RNN, $score(b)$ is the accumulated score of the already-generated words in beam state b, $f_i(w)$ is the ith feature function, and w_i is the corresponding weight. The likelihood of words with desired feature in vocabulary V_{suc} of the succeeding state is thus augmented by this scoring function during decoding process.

Baheti et al. [1] proposed two constraints(topic and semantic constraint) for use in decoding objective, which help generate more content rich responses. When incorporating these constraints during decoding(Beam Seach), they used an additive variants of standard methods that factorize over words.

2) Sampling:

a) Temperature Sampling:

Holtzman et al. [12] introduced the approach of temperature sampling. A temperature t is applied to softmax for changing the probability distribution of vocabulary to skew it towards high probability events:

$$P(x|x_{1:t-1}) = \frac{exp(u_t/t)}{\sum_{t'} exp(u_{t'}/t)}$$

where $u_{1:|V|}$ is the logits given and $t \in [0, 1)$.

Temperature sampling has been widely applied to text generation. However, the author also mentioned that although low temperature would increase quality of generation, it might cause repetition and low diversity within text.

b) Top-k Sampling:

Fan et al. [6] introduced top-k sampling method, which selects the top-k most likely next words at each time step and the probability mass is redistributed among them. Therefore, k words would be filtered for a generation in total. Top-k Sampling could generate less repetitive text and is adopted by generation tasks like GPT-2. The limitation for top-k sampling is that k words should be pre-defined, without dynamic consideration.

c) Nucleus Sampling:

Holtzman et al. [12] introduced Nucleus sampling(top-p) sampling to address the issue of top-k sampling. However, Nucleus Sampling focus on the smallest set of words with cumulative probability exceeds the probability p. Then the original probability is re-scaled to sample new tokens.

3) Loss:

a) Classifier Loss:

Classifier has been widely applied to style transfer task as a way for generating samples with desired style. Prabhumoye et al. [24] trained a convolutional neural network classifier to predict the given style and evaluate samples. The classifier is

trained in a supervised manner to accept the generator output as input:

$$\mathcal{L}_{class}(\theta_C) = \mathbb{E}_X \left[\log q_C(s \mid x)\right]$$

where X is the dataset with label $s(s_0$ or $s_1)$, and θ_C is the classifier parameter.

Prabhumoye et al. [25] used a similar classifier training for generating stories based on five different personas. However, when the number of target styles increases, the accuracy of the classifier decrease, so the model might perform poorly at test time.

Sudhakar et al. [33] introduced Generative Style Transformer(GST) that rewrite sentences to a target style in the absence of parallel style corpora. It used an unsupervised-learning method to train for DELETEONLY and DELETEANDRETRIEVE model. For the DELETEONLY loss as instance:

$$\mathcal{L}(\theta) = \sum_{(x,v^{src})\in\mathcal{D}} \log p(x \mid c(x, v^{src}), a'(x, v^{src}); \theta)$$

The loss could help guide the model to reconstruct text with desired styles given original content and attributes.

Hu et al. [14] also used a classifier based approach for visual story generation. They provided a versatile toolkit, Taxer, for multiple text generation tasks. A pre-trained classifier is used as evaluation for transfer style accuracy.

b) Strategy Loss:

Zhou et al. [48] utilized a strategy base approach for negotiation text generation. An intermediate step of strategy prediction is added before generation to predict the possible strategies. The strategy loss for this task is:

$$\begin{aligned}\mathcal{L}_{ST} = &- \sum_{\{j|st'_{t+1,j}=1\}} \log(st_{t+1,j}) \\ &- \sum_{\{j|st'_{t+1,j}=0\}} \log(1 - st_{t+1,j})\end{aligned}$$

where $st_{t+1,j}$ is the possible strategies and $st'_{t+1,j}$ is the ground truth strategies.

By combining the strategy prediction task, system utterance generation loss together, and a posterior constraint to get a joint loss, the joint loss could guide the generator for generating text with target strategies.

c) Coverage Loss:

For the summarization task, the coverage proerties are used for calculating loss and guide generation.

See et al. [31] used coverage to keep track of summarized text to discourage repetition. The coverage loss is defined to penalize repeatedly attending to the same locations, and thus avoid generating repetitive text:

$$covloss_t = \sum_i min(a_i^t, c_i^t)$$

where c is the coverage vector and a is the attention distributions. The coverage loss is reweighted and added to the primary loss function.

Li et al. [20] imported structural-compression and structural-coverage regularization into summarization process in order to capture the information compression and information coverage properties. The author included these two loss objective, structural-compression and structural-coverage at sentence-level attention.

$$\begin{aligned}\mathcal{L} = \sum_{(X,Y)\in\tau} \Big\{ &- \log P(Y \mid X; \theta) \\ &+ strComLoss(\alpha_t) + strConvLoss(\alpha_t) \Big\}\end{aligned}$$

The loss function $\mathcal{L}$ combines the negative log-likelihood of generating summaries over training set τ, the structural-compression loss and structural-coverage loss, where α_t is the sentence-level attention distribution when generating the tth summary sentence .

IV. Open Problem

Although there exist various current researches related to controllable text generation, the topic still needs more exploration. Based on this review paper, it could be observed that different methods still have limitations, for instance, the PPLM model might be expensive for computation, and VAE is weak in the ability for specific information control, etc. The decision of some researches to improve the model performance and the quality of generated text is to combine different methods together, for instance, Zhou et al. [48] utilized both GRU and strategy loss for dialogue generation with target strategies, and Hu et al. [13] utilized VAE and classifier loss for sentence generation with target attributes. The decisions might vary depending on the tasks and goals. However, there is still lack of an effective method for evaluating the and determining the best combination methods. Thus, an automatic evaluation still remains as an open problem.

V. Conclusion and Future Work

In this review project I tend to classify the research papers related to controllable text generation. I aim for forming a systematic understanding by dividing the paper collections into six main catalogs based on the most frequent approaches researches have used: Variational Auto-Encoder(VAE), Recurrent Neural Network(RNN), Pre-trained Language Model(PLM), Attention and Transformer, Fine-tuning, and Training Objective. I also included detailed explanations and analysis for the techniques and model in each paper with evaluations, as well as a summary for its content and research goal. The future work for this review project is to extend this classification schema by considering more research papers and make more evaluation between papers by making empirical analysis and parallel comparisons.

References

[1] Ashutosh Baheti, Alan Ritter, Jiwei Li, and Bill Dolan. Generating more interesting responses in neural conversation models with distributional constraints, 2018.

[2] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. Electra: Pre-training text encoders as discriminators rather than generators, 2020.

[3] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1edEyBKDS.

[4] Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. Wizard of wikipedia: Knowledge-powered conversational agents, 2019.

[5] Yu Duan, Canwen Xu, Jiaxin Pei, Jialong Han, and Chenliang Li. Pre-train and plug-in: Flexible conditional text generation with variational auto-encoders, 2020.

[6] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation, 2018.

[7] Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. Style transfer in text: Exploration and evaluation, 2017.

[8] Andrea Galassi, Marco Lippi, and Paolo Torroni. Attention in natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, page 1–18, 2020. ISSN 2162-2388. doi: 10.1109/tnnls.2020.3019893. URL http://dx.doi.org/10.1109/TNNLS.2020.3019893.

[9] Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. Hafez: an interactive poetry generation system. In *Proceedings of ACL 2017, System Demonstrations*, pages 43–48, Vancouver, Canada, July 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P17-4008.

[10] Hongyu Gong, Suma Bhat, Lingfei Wu, Jinjun Xiong, and Wen mei Hwu. Reinforcement learning based text style transfer without parallel training corpus, 2019.

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

[12] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration, 2020.

[13] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. Toward controlled generation of text, 2018.

[14] Zhiting Hu, Haoran Shi, Bowen Tan, Wentao Wang, Zichao Yang, Tiancheng Zhao, Junxian He, Lianhui Qin, Di Wang, Xuezhe Ma, Zhengzhong Liu, Xiaodan Liang, Wangrong Zhu, Devendra Singh Sachan, and Eric P. Xing. Texar: A modularized, versatile, and extensible toolkit for text generation, 2019.

[15] Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation, 2019.

[16] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014.

[17] Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. Gedi: Generative discriminator guided sequence generation, 2020.

[18] Ilia Kulikov, Alexander H. Miller, Kyunghyun Cho, and Jason Weston. Importance of search and evaluation strategies in neural dialogue modeling, 2019.

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

[20] Wei Li, Xinyan Xiao, Yajuan Lyu, and Yuanzhuo Wang. Improving neural abstractive document summarization with structural regularization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4078–4087, 2018.

[21] Fuli Luo, Peng Li, Pengcheng Yang, Jie Zhou, Yutong Tan, Baobao Chang, Zhifang Sui, and Xu Sun. Towards fine-grained text sentiment transfer. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2013–2022, Florence, Italy, 2019. Association for Computational Linguistics.

[22] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization, 2017.

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

[24] Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 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 2018. Association for Computational Linguistics.

[25] Shrimai Prabhumoye, Khyathi Raghavi Chandu, Ruslan Salakhutdinov, and Alan W Black. "my way of telling a story": Persona based grounded story generation, 2019.

[26] Shrimai Prabhumoye, Alan W Black, and Ruslan Salakhutdinov. Exploring controllable text generation techniques, 2020.

[27] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.

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

[29] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks, 2016.

[30] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec

Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[31] Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks, 2017.

[32] Louis Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope, and Ray Kurzweil. Generating high-quality and informative conversation responses with sequence-to-sequence models, 2017.

[33] Akhilesh Sudhakar, Bhargav Upadhyay, and Arjun Maheswaran. "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)*, Hong Kong, China, November 2019. Association for Computational Linguistics.

[34] Jack Urbanek, Angela Fan, Siddharth Karamcheti, Saachi Jain, Samuel Humeau, Emily Dinan, Tim Rocktäschel, Douwe Kiela, Arthur Szlam, and Jason Weston. Learning to speak and act in a fantasy text adventure game, 2019.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[36] Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R. Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. Diverse beam search: Decoding diverse solutions from neural sequence models, 2018.

[37] Wenlin Wang, Zhe Gan, Hongteng Xu, Ruiyi Zhang, Guoyin Wang, Dinghan Shen, Changyou Chen, and Lawrence Carin. Topic-guided variational autoencoders for text generation, 2019.

[38] Ziwen Wang, Jie Wang, Haiqian Gu, Fei Su, and Bojin Zhuang. Automatic conditional generation of personalized social media short texts. *PRICAI 2018: Trends in Artificial Intelligence*, page 56–63, 2018.

[39] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. Semantically conditioned lstm-based natural language generation for spoken dialogue systems, 2015.

[40] Lilian Weng. Controllable neural text generation, 2021. URL https://lilianweng.github.io/lil-log/2021/01/02/controllable-neural-text-generation.html.

[41] Zeqiu Wu, Michel Galley, Chris Brockett, Yizhe Zhang, Xiang Gao, Chris Quirk, Rik Koncel-Kedziorski, Jianfeng Gao, Hannaneh Hajishirzi, Mari Ostendorf, and Bill Dolan. A controllable model of grounded response generation, 2020.

[42] Peng Xu, Jackie Chi Kit Cheung, and Yanshuai Cao. On variational learning of controllable representations for text without supervision, 2020.

[43] Jiang Xiang Men Ya. from attention to transformer, 2019. URL https://www.jianshu.com/p/c6e74e8b4379.

[44] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding, 2020.

[45] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, 2016.

[46] Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. Personalizing dialogue agents: I have a dog, do you have pets too?, 2018.

[47] Yizhe Zhang, Guoyin Wang, Chunyuan Li, Zhe Gan, Chris Brockett, and Bill Dolan. Pointer: Constrained progressive text generation via insertion-based generative pre-training, 2020.

[48] Yiheng Zhou, Yulia Tsvetkov, Alan W Black, and Zhou Yu. Augmenting non-collaborative dialog systems with explicit semantic and strategic dialog history. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ryxQuANKPB.

[49] Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences, 2020.