A Method for Generating Synthetic Electronic Medical Record Text

Jiaqi Guan, Runzhe Li, Sheng Yu, and Xuegong Zhang

Abstract—Machine learning (ML) and Natural Language Processing (NLP) have achieved remarkable success in many fields and have brought new opportunities and high expectation in the analyses of medical data, of which the most common type is the massive free-text electronic medical records (EMR). However, the free EMR texts are lacking consistent standards, rich of private information, and limited in availability. Also, it is often hard to have a balanced number of samples for the types of diseases under study. These problems hinder the development of ML and NLP methods for EMR data analysis. To tackle these problems, we developed a model called Medical Text Generative Adversarial Network or mtGAN, to generate synthetic EMR text. It is based on the GAN framework and is trained by the REINFORCE algorithm. It takes disease tags as inputs and generates synthetic texts as EMRs for the corresponding diseases. We evaluate the model from micro-level, macro-level and application-level on a Chinese EMR text dataset. The results show that the method has a good capacity to fit real data and can generate realistic and diverse EMR samples. This provides a novel way to avoid potential leakage of patient privacy while still supply sufficient well-controlled cohort data for developing downstream ML and NLP methods.

Index Terms—Synthetic electronic medical record text, conditional model, generative adversarial network, reinforcement learning

1 Introduction

The widespread adoption of Electronic Medical Records (EMR) has brought new opportunities in the biomedical domain, and clinical narratives are significant components of EMRs. Due to relevant laws and regulations for the protection of patient privacy, EMR data are generally inaccessible to the majority of the ML community. In addition, when studying a disease, the positive and negative EMR samples are usually highly imbalanced, which makes it difficult to train ML algorithms with real medical records. To prevent potential leakage of patient privacy, as well as to provide a sufficient data source for machine learning, we aim to develop a model to generate synthetic textual EMR datasets.

A general way to alleviate the privacy risks is via de-identification, which is the process of reducing the information associated with an individual's identity. The anonymization is typically done by applying generalization and suppression operations to modify the patients' attributes [1]. However, these approaches cannot fully avoid privacy disclosure, since the anonymous patients can be re-identified using specific information [2]. Generating synthetic text of medical records is a way to completely avoid possible re-identifications. Text generation is one of the most fundamental problems in natural language processing. During the recent decade, deep neural networks have achieved remarkable success in several tasks and researchers are paying more attention to the text generation via deep learning models. A promising approach to text generation is training a recurrent neural network (RNN) by maximum likelihood estimation (MLE) [3]. However, it suffers from the well-known *exposure bias* [4] problem. The success of Generative Adversarial Network (GAN) [5] has inspired researchers to investigate adversarial training over textual data, while another new problem has arisen when using GAN to generate discrete data: the gradient cannot be back-propagated from the discriminator to the generator. Some related work (such as SeqGAN [6]) utilize the REINFORCE algorithm [7], which is a classical policy gradient algorithm in reinforcement learning, to optimize the original GAN objective.

Our proposed model Medical Text Generative Adversarial Network (mtGAN) is a GAN-based framework and we adopt the REINFORCE algorithm to train the model. To satisfy different demands of research, our mtGAN is a conditional model with designated disease tags as inputs, and can generate corresponding EMR text data. We test two discriminative models (CNN, BiRNN with attention mechanism) and different methods to rescale rewards to achieve the best performance. We design micro-, macro- and application-level experiments to demonstrate the effectiveness of our model. In the micro-level experiment, our model has the strong ability to fit the real data and generate diverse examples at the same time. In the macro-level experiment, our model has the best adversarial success in the adversarial evaluation. In the application-level experiment, we design a disease classification experiment and the results suggest that the synthetic

- *J. Guan is with the Department of Computer Science, Univeristy of Illinois Urbana-Champaign, Champaign, IL 61820 USA. E-mail: guanjq14@tsinghua.org.cn.*
- *R. Li is with the Department of Biostatistics, Johns Hopkins University 1466, Baltimore, MD 21218 USA. E-mail: lrz14@tsinghua.org.cn.*
- *S. Yu is with the Center for Statistical Science, Institute for Data Science, Department of Industrial Engineering, Tsinghua University, Beijing 100084, China. E-mail: syu@tsinghua.edu.cn.*
- *X. Zhang is with the Department of Automation, Tsinghua University, Beijing 100084, China, the MOE Key Laboratory of Bioinformatics and Bioinformatics Division, Beijing National Research Centre for Information Science and Technology (BNRIST), Beijing 100084, China, and the Center for Synthetic and Systems Biology, Tsinghua University, Beijing 100084, China. E-mail: zhangxg@tsinghua.edu.cn.*

Manuscript received 30 Mar. 2019; revised 21 Sept. 2019; accepted 2 Oct. 2019. Date of publication 23 Oct. 2019; date of current version 3 Feb. 2021. (Corresponding author: Jiaqi Guan.)
Digital Object Identifier no. 10.1109/TCBB.2019.2948985

EMR texts generated by our model are capable of producing comparable properties to real data. Therefore our synthetic datasets can also be used as a data augmentation approach for machine learning tasks.

2 Related Work

Text generation has been one of the most challenging problems in natural language processing. Recurrent Neural Network (RNN) and its variants Long Short-Term Memory (LSTM) [8] and Gated Recurrent Unit (GRU) [9] have achieved impressive performance in several complex tasks, such as machine translation and dialogue generation[10]. The RNN language models are commonly used for sequence generation, and they are trained by maximum likelihood estimation (MLE) in an approach called *teacher forcing* [11]. This mainstream method for auto-regressive models predicts the next token given the previous ground-truth tokens, which leads to the exposure bias problem.

Generative Adversarial Network (GAN) proposed by Goodfellow provides an alternative framework to generate synthetic data. The GAN model consists of two neural networks: a generator G trying to generate synthetic data, and a discriminator D trying to distinguish the real data from the synthetic. The training procedure is a two-player zero-sum game between G and D. GANs have enjoyed great success in image generation, but they are not as much widely applied in natural language processing tasks. One reason is that the gradient from the discriminator cannot be back-propagated to the generator due to the discrete outputs. To address this problem, Yu proposed seqGAN [6], where the generator is updated through the policy gradient using Reinforcement Learning (RL), and the reward is calculated by the discriminator on a complete sequence via Monte Carlo search. However, no one has ever used this kind of approach in the case of generating synthetic EMR text.

Recent studies attempt to generate synthetic electronic medical records via deep generative models. For example, Choi proposed a new model medGAN to generate realistic EMRs with high-dimensional binary and count variables [12]. Hyland proposed a Recurrent (Conditional) GAN to generate real-valued time series in medical application [13]. Yahi utilized a GAN framework to produce continuous time series data in EMRs, which can predict the effects of drug exposure [14]. However, most of the research on synthetic medical records are based on highly structured data formats, including numerical and categorical variables. The majority of clinical documents are saved in unstructured textual formats, and it is labor intensive to process the raw medical texts before the analysis. Under such circumstances, we propose a model called *mtGAN* for the generation of synthetic EMR text. The primary contributions of our paper are listed as follows:

- We propose a model called mtGAN to generate synthetic EMR text.
- We can control the generation process with assigned specific disease tags to satisfy different research demands.
- We design micro-level, macro-level and application-level evaluation methods to assess the model performance, and our model outperforms other baseline models.
- The application-level experiment results demonstrate that our synthetic data can achieve similar performance in machine learning tasks compared with the real data, which can be used as a data augmentation approach then.

3 Preliminaries

3.1 Synthetic EMR Text Generation Problem

The synthetic EMR text generation problem is fundamentally a discrete sequence generation problem. Specifically, suppose we have a set of real-world EMR data $S^+ = \{X^i\}_{i=1}^N$, where each data consists of a sequence of words $X = \{x_1, x_2, \ldots, x_T\}$ and each word comes from a vocabulary of candidate tokens. Our goal is to produce a set of EMR data which have similar characteristics to real-world EMR data by learning the underlying distribution of the real data p_d, so that it can be used in more cases to substitute privacy-sensitive and limited real EMR data.

3.2 Recurrent Neural Networks

In recent years, neural networks have shown remarkable results in the text generation task. Among all neural network structures, Recurrent Neural Network (RNN) and its improved variants, such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), are most widely adopted given their promising capacity to capture long-term dependencies in the text. RNN is a language model with Markov property, i.e., at each time step, it will encode all previous inputs to a hidden vector h_t and will use it to conduct the inference of the next token. This procedure can be formulated as:

$$\begin{cases} h_t = f(h_{t-1}, x_t) \\ o_t = g(h_t) \end{cases}. \tag{1}$$

3.3 Generating with Maximum Likelihood Estimation

In the text generation task, the general method to train a language model is through Maximum Likelihood Estimation (MLE). The MLE optimization method regards the original text generation problem as a sequential multi-label classification problem at all time steps, which converts the original unsupervised learning task to a supervised learning task. For a RNN generator G_θ, the MLE objective is to minimize the multi-label cross entropy, which can be formulated as (2). For the simplicity of notations, we will also denote the probability of generating tokens as $G_\theta(\cdot|\cdot)$.

$$J_G(\theta) = \mathbb{E}_{X \sim p_d}\left[-\sum_{t=1}^{T} \log G_\theta(x_t | X_{1:t-1})\right]. \tag{2}$$

3.4 Generating With Adversarial Reinforcement Learning

Although MLE has better convergence performance and training robustness than other algorithms, it suffers from the *exposure bias* problem, which makes MLE less useful in generating long texts. To tackle this problem, recent works focus on generating texts under the Generative Adversarial

Network (GAN) setting. In the standard GAN setting, there is a generator G that plays a minimax game against a discriminator D. The generator G transforms a noise z sampled from a noise distribution p_z to a data sample $G(z)$, and tries to match the generated distribution p_g to the real data distribution p_d. The discriminator D is a binary classifier, which takes real data as positive samples while synthetic data as negative samples, and tries to distinguish them to give the generator training signals. The two-player minimax game for continuous data can be formulated as follows, where $G(\cdot)$ denotes the generated sample, $D(\cdot)$ denotes the probability given by D that the sample comes from the real distribution.

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_d(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]. \tag{3}$$

It can be seen that the standard GAN framework requires the generated data is differentiable so that the gradient can back-propagate from the discriminator to the generator to update parameters of the model. This constraint makes it not trivial to apply the standard GAN framework to discrete data, such as EMR text data. One way to solve this problem is to apply a typical reinforcement learning (RL) algorithm to the generator. Text generation with RNNs can be viewed as a sequential decision process. Hence, we can model the generator as a policy of picking the next token. The key elements in RL are as follows:

1) State: The generated tokens so far $x_{1:t-1}$
2) Action: The next token to be generated x_t
3) Reward: The GAN discriminator's output, i.e., the likelihood that the synthetic sentence can fool the discriminator, which indicates how good the entire sentence is.

The reward obtained by the aforementioned way will be used for all actions (the generated tokens), but not for intermediate actions separately. However, in some cases, the discriminator might assign a low reward due to some part of the generated sentence, which is not ideal for the good generated parts. Thus, rewards for intermediate generation steps are necessary. One simple strategy to compute intermediate rewards is using Monte Carlo (MC) search [15]. In Monte Carlo search, given a partially generated sentence $X_{1:t-1}$, the model keeps sampling tokens from the current distribution until the sentence finishes, and repeats this sampling procedure for K times. These K samples are fed to the discriminator, and the average score is used as the reward for x_t.

With well-defined RL elements and intermediate rewards, we can optimize the model with the REINFORCE (policy gradient) algorithm. Given a generator G_θ which tries to maximize the rewards it receives from the discriminator, and a discriminator D_ϕ which still tries to distinguish real text from synthetic text, the objective can be formulated as (4) and (5). For generating discrete data, $G(x_t|X_{1:t-1})$ denotes the probability of generating token x_t given previous generated tokens $X_{1:t-1}$.

$$\max J_G(\theta) = \mathbb{E}_{X\sim G_\theta}\left[\sum_{t=1}^{T} \log G_\theta(x_t|X_{1:t-1}) \cdot R_{D_\phi}^{G_\theta}(X_{1:t-1}, x_t)\right] \tag{4}$$

$$\max J_D(\phi) = \mathbb{E}_{X\sim p_d}[\log D_\phi(X)] + \mathbb{E}_{X\sim G_\theta}[\log(1 - D_\phi(X))], \tag{5}$$

where,

$$R_{D_\phi}^{G_\theta}(s = X_{1:t-1}, a = x_t) = \frac{1}{K}\sum_{k=1}^{K} D_\phi(X_{1:T}^k), X_{1:T} \in \mathrm{MC}(X_{1:t}). \tag{6}$$

Our synthetic EMR text generation model will base on the GAN framework and use REINFORCE as our optimization algorithm.

4 Method

4.1 Medical Text Generative Adversarial Network

To apply GAN to generate synthetic EMR text data, we propose a conditional GAN framework named Medical Text Generative Adversarial Network (mtGAN). In the medical domain, EMR text usually consists of disease descriptions and diagnostic results. The underlying mapping from disease descriptions to diagnostic results is what researchers devote lots of time and energy to explore. However, on one hand, researchers sometimes only care about some specific diagnostic results, but the related disease descriptions usually mix with redundant information. On the other hand, samples for specific diseases are scarce, which gives rise to the pressing need to generate EMR text data with specific diagnostic results, or disease tags. Our proposed mtGAN is a conditional model [16] that takes designated disease tags as the conditional constraint of inputs, and generates corresponding EMR text consistent and focused on the given input feature. For example, the disease tags can be pneumonia or lung cancer, which indicates the severity of the disease. When we train the mtGAN model, these features can be extracted from complete EMRs. The overall mtGAN model is shown in Fig. 1a. We input disease tags to guide the generator to produce corresponding synthetic disease descriptions. The discriminator is trained to distinguish real EMR text and synthetic EMR text. The classification results of discriminator are also reward signals to guide the training of generator with policy gradient algorithm. A sketch of the training of mtGAN is shown in Algorithm 1. It is worth noting that the introduction of conditional constraint is the main difference compared to seqGAN. More details are described as follows.

Given a generator G_θ parameterized by θ and a discriminator D_ϕ parameterized by ϕ, we introduce an additional conditional constraint y based on the original GAN framework. The original two-player minimax game becomes a game with a conditional probability form.

In practice, we also apply REINFORCE [7] to solve this problem. The objectives of the generator and the discriminator can be rewritten as follows:

$$\max \mathbb{E}_{X\sim G_\theta(\cdot|y)}\left[\sum_{t=1}^{T} \log G_\theta(x_t|X_{1:t-1}, y) \cdot R_{D_\phi}^{G_\theta}(X_{1:t-1}, x_t, y)\right] \tag{7}$$

$$\max \mathbb{E}_{(X,y)\sim p_d}[\log D_\phi(X,y)] + \mathbb{E}_{X\sim G_\theta(\cdot|y)}[\log(1 - D_\phi(X,y))], \tag{8}$$

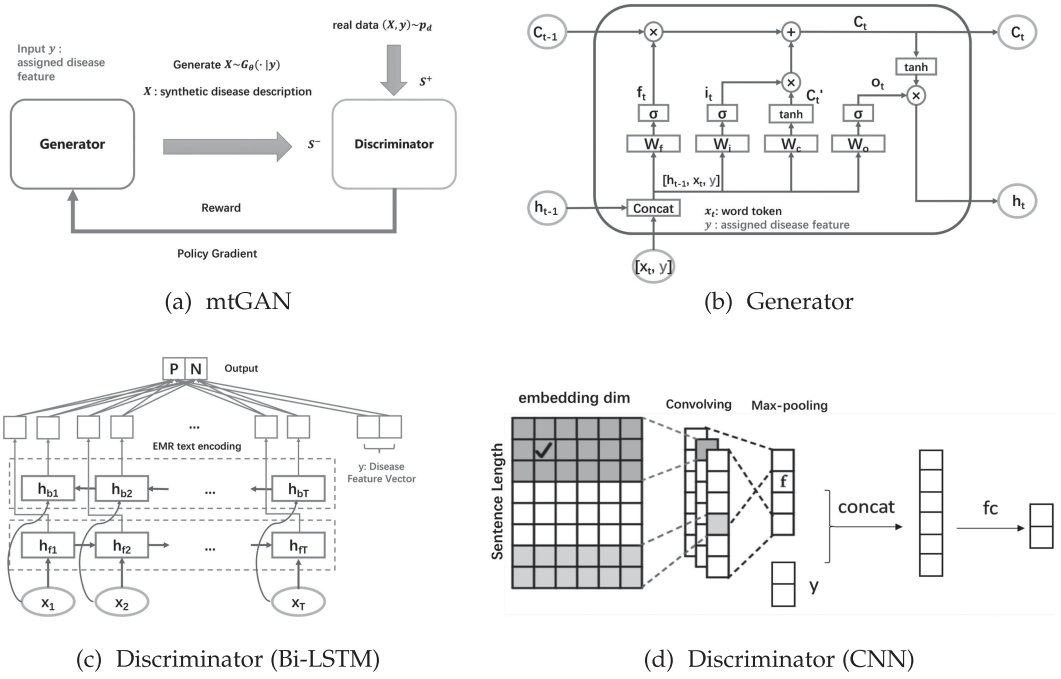


Fig. 1. *Our model mtGAN*. We input disease tags to guide the generator to produce corresponding synthetic disease descriptions. The model is trained with policy gradient algorithm. The second row shows two discriminator structures. (d) is modified from Fig. 6 in [21].

where,

$$R_{D_\phi}^{G_\theta}(s = X_{1:t-1}, y, a = x_t) = \frac{1}{K}\sum_{k=1}^{K} D_\phi(X_{1:T}^k, y), \quad X_{1:T} \in \mathrm{MC}^{G_\theta}(X_{1:t}, y). \tag{9}$$

4.2 Generator Specification

We use Long Short-Term Memory (LSTM), an improved variant of recurrent neural network, as our generative model. An RNN maps the input word embedding representation x_t to the hidden state h_t with an update function f recursively, i.e., $h_t = f(h_{t-1}, x_t)$. The improvement of LSTM is that it uses three well-designed gates to implement f. These gates all include one neural network layer with Sigmoid activation function to output a number between 0 to 1, which controls how much information can pass to the cell state C_t. Specifically, the Forget Gate decides how much information to be abandoned from the cell state; the Input Gate decides what information will be saved from the cell state; the Output Gate filters the input and decides the new hidden state. To impose the conditional constraint, the disease tags are fed into the model as an additional input at every generation step. The whole procedure can be formulated as:

$$\begin{cases} f_t = \sigma(W_f \times [h_{t-1}, x_t, y] + b_f) \\ i_t = \sigma(W_i \times [h_{t-1}, x_t, y] + b_i) \\ \tilde{C}_t = \tanh(W_C \times [h_{t-1}, x_t, y] + b_C) \\ C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \\ o_t = \sigma(W_o \times [h_{t-1}, x_t, y] + b_o) \\ h_t = o_t \cdot \tanh(C_t) \end{cases}, \tag{10}$$

where f_t denotes the output of Forget Gate, i_t denotes the output of Input Gate, o_t denotes the output of Output Gate, $W(\cdot)$ and $b(\cdot)$ are weight and bias parameters of LSTM.

Algorithm 1. Medical Text Generative Adversarial Network

Require: Generative network G_θ; Rollout network G_β and update rate α; Discriminative network $D\phi$; Real EMR text dataset $S^+ = \{X_{1:T}, y\}$

Initial G_θ, D_ϕ with random weights θ, ϕ
Pre-train G_θ using MLE on S^+
$\beta \leftarrow \theta$
Generate samples S^- using G_θ for training D_ϕ
Pre-train D_ϕ by (8) on S^+, S^-
for total iterations **do**
  for g-steps **do**
    Assign random disease tags y, generate sequences $X_{1:T} = (x_1, \ldots, x_T) \sim G_\theta(X|y)$
    for t in $1:T$ **do**
      Compute rewards $R_{D_\phi}^{G_\beta}$ by (9)
    end for
    Update generator parameters via (7)
  end for
  for d-steps **do**
    Generate negative examples S^- using current G_θ and combine with given positive examples S^+
    Update discriminator parameters via (8) for k epochs
  end for
  $\beta \leftarrow (1-\alpha)\theta + \alpha\beta$
end for

It is worth noticing that other RNN variants, such as GRU, can also be used the generative model. A typical LSTM generator is shown in Fig. 1b.

4.3 Discriminator Specification

The discriminator needs to execute a two-class text classification task. We test two types of classifiers in the experiment section: convolutional neural network (CNN) and bidirectional recurrent neural network (BiRNN) with attention mechanism. It is worth noticing that the discriminator

has a crucial impact on the final generation results, as it provides reward signals to guide the update of the generator.

CNN. Convolutional neural network is the main framework for solving computer vision problems, but it has also shown great performance in text classification recently [17]. Given an input sentence $X = \{x_1, \ldots, x_T\}$, we also embed each word to a vector representation first, and the input can be represented as a 2-dimensional matrix $\varepsilon_{1:T} \in \mathbb{R}^{T \times k}$, where k is the dimension of word embeddings. Then, we perform convolutional operations on the sentence matrix with different sizes and numbers of filters, of which the second dimension is always k. For a filter with size h, we can get a feature map $\tilde{c} = \{c_1, \ldots, c_{T-h+1}\}$, where $c_i = f(\mathbf{w} \otimes \varepsilon_{i:i+h-1} + b)$. Finally we perform max-pooling on $\tilde{c}$ to get $\hat{c} = \max\{\tilde{c}\}$ and fully-connected on $\hat{c}$ to get the final result. Similar to the discriminator in SeqGAN, we also use a residual highway structure before final fully-connected layers to enhance the predictive performance.

BiRNN-Attention. Recurrent neural network should be the most direct model to conduct text classification task. We can simply use hidden states of the last time step to predict labels. In practice, we use the bidirectional LSTM structure and the attention mechanism to enhance the performance [18]. The bidirectional structure makes the output of each time step is the contribution of current input word to both above and following contexts, rather than only above contexts. The attention mechanism considers that different time steps have different contributions to the target task. Thus, it assigns normalized weights to the hidden states of each time step, and then uses the weighted sum of hidden states to predict. The attention mechanism can be formulated as:

$$\begin{cases} u_t = \tanh(W_w h_t + b_w) \\ \alpha_t = \frac{\exp(u_t^T u_w)}{\Sigma_t \exp(u_t^T u_w)} \\ s = \sum_t \alpha_t h_t \end{cases}, \tag{11}$$

where W_w and b_w denotes weights and bias of the attention layer, u_w denotes weights of the fully-connected layer to compute α. α_t is the softmax-output of the fully-connected layer, which represents the weights of outputs at each time step, and s is the final output.

Among all discriminators, the conditional constraint is fed into the final fully-connected layer as an additional input.

4.4 Approaches to Stable Adversarial Training

In the adversarial text generation, the training usually suffers from two main problems [19]. One is the gradient vanishing problem, which means that if the discriminator is much stronger than the generator, the generated samples will always obtain almost 0 reward, which causes updates of the generator to nearly stop. The other is the mode collapse problem, which is caused by the REINFORCE algorithm. The generator usually tends to produce short repeated parts to earn high evaluation from the discriminator, which makes the overall quality and diversity of generated samples pretty low. To alleviate these problems, we use several approaches to enhance the stability of adversarial training.

Rescale Rewards. To alleviate the gradient vanishing problem, a straight method is to use rescaled scores as reward signals. In the experiment section, we test two rescaled methods. The first one is proposed in MaliGAN [20]: $R = \frac{D}{1-D}$, and we call it ODA (Optimal Discriminator Activation) for short. The second one is BRA(Bootstrapped Ranking Activation), which is proposed in LeakGAN [21]: $R = \sigma(\delta \cdot (0.5 - \frac{\text{rank}(i)}{B}))$, where rank$(\cdot)$ denotes the sequence's high-to-low ranking in the batch, δ is the activation smoothness hyperparameter, B is the batch size, and $\sigma(\cdot)$ is a non-linear function. ODA assumes that the discriminator is optimal, while BRA uses the ranking information in each batch to rescale rewards. These two methods do not require any modification on the model structure. Furthermore, we can introduce an action-independent baseline b (usually using the mean reward in each batch), and use $R - b$ to replace R to guide the update of the generator's parameters.

Teacher Forcing. Considering that the generator does not get access to real samples directly, we conduct one step of teacher forcing (MLE) after one step of adversarial training. In fact, the difference between teacher forcing and REINFORCE is that teacher forcing uses texts from real data and the value of rewards is 1, while adversarial training uses texts generated by G_θ are rewards are given by discriminator. In practice, we find teacher forcing can effectively alleviate the mode collapse problem.

Delayed Rollout Network. From the RL viewpoint, the generator can be viewed as a *actor* and the discriminator can be viewed as a *critic* [22]. To improve the training convergence and stability, we build up a copy of generator G_β that is soft-updated with G_θ to sample sentences in MC search. This delayed rollout network has smaller changes in parameters, which makes the training more stable.

5 Experiments

In this section, we first describe the EMR text dataset that we use. Then, we describe the implementation details of our model. To ensure that mtGAN achieves the best performance, we conduct experiments with different rescale methods and discriminator structures. To test the effectiveness of our model, we design the micro-level, macro-level and application-level experiments, from which we can see mtGAN has better performance than the baselines.

5.1 Dataset

The EMR texts that we use in the experiment section are in Chinese, but it is worth noticing that the difference between EMR texts in different languages only lies in the approaches of data pre-processing. After EMR texts are converted into sequences of word embedding vectors, we can simply apply our model to them.

We collected 2216 EMR texts from the respiration department of a hospital to construct the dataset. Original EMR texts include personal information, chief complaint, history of present illness, history of past illness, admission diagnosis and so on. To protect the privacy of patients, we remove sensitive information such as person names and place names. They are removed according to the format of our collected EMR text or auto-removed during the pre-processing due to the low frequencey of appearance. It is worth mentioning that traditional methods including how we remove the privacy information suffers the risk of re-identification, while our generative model can effectively solve this problem.

TABLE 1
Examples of Original Real EMR Text

Type	Examples
Pneumonia	患者 病情 平稳， 偶有 咳嗽， 咳痰， 为 白色 痰， 无 明显 喘 憋， 无 痰中 带血， 无发热 。 今日 为行 进一步 气管镜 下 治疗 入院。 发病 以来， 神清 清楚， 精神 可 The patient's condition is stable, sometimes has a cough, sputum, which is white phlegm, no obvious asthma and suffocation, no phlegm with blood, and no fever. Be admitted to the hospital today for further bronchoscope treatment. Since the onset of the disease, the spirit is clear.
Lung Cancer	患者 1 年 多 前 出现 咳嗽， 咳痰 费力， 无 明显 气促， 在 外院 考虑 为 " 支气管炎 "， 给予 口服 头孢类 抗生素， 服药 后 疗效 不 佳， 行 胸部 CT 示 右肺 占位 。 The patient developed a cough more than a year ago, coughing phlegm laboriously, no obvious shortness of breath, in the hospital considered as "bronchitis," given oral cephalosporin antibiotics, the efficacy was poor, chest CT showed that the right lung occupied space.

TABLE 2
Examples of Generated Synthetic EMR Text

Type	Examples
Pneumonia	患者于1周前无明显诱因出现咳嗽，咳白色粘痰，伴活动后加重，休息后可缓解，间断服 用镇咳药物等治疗，未行正规诊治。 The patient had no obvious cause of cough a week ago, coughing white phlegm, aggravated after activity, can be relieved after rest, intermittently use antitussive drugs and others for treatment, no formal diagnosis and treatment.
Lung Cancer	患者10多年前开始出现咳嗽、咳痰，痰中带血，当地医院查胸部CT示纵隔肿大淋巴结，右肺下叶结节，后行气管镜时治疗收入院。 The patient began to cough and sputum more than 10 years ago, with blood in the phlegm. Chest CT showed mediastinal enlarged lymph nodes and nodules in the right lower lobe of the lung. Be admitted to the hospital after endoscopic treatment.

We use history of present illness as input sequences and admission diagnosis as sequence tags. In practice, we use two tags as the conditional constraint of generation: pneumonia and lung cancer. This is because the disease descriptions are mainly about lung disease. Some patients may have other medical history, such as heart disease and hypertension, but they are more dispersed which can not be used to train our model. However, our model can also be easily extended to multiple tags with adjusting the dimension of conditional constraint input and encoding multiple tags. We segment words in each EMR text with the jieba package [23] with an additional medical dictionary. Numbers are completely split, words even with low frequency are kept and word-level segmentation is adopted instead of character-level segmentation. We find these pre-processing operations can improve the readability of generated EMR texts to some extent in practice. In the end, the dictionary of this dataset includes 7674 words.

We cut the first 40 words of each EMR text as the input to the model for convenience, as it will not take a long time to train the model but can also generate informative results. If the length of text is too long, the long-term dependence relationship of text becomes complicated and the quality of generated text can not be guaranteed. Long EMR text generation will be our future research goal. After pre-processing about 97.7 percent samples have the length of 40, others have less than 40 words. After removing invalid and repeated data, the dataset is split into training, validation and test set with the proportion of 0.7, 0.1, 0.2 separately. The ratio of samples with pneumonia tag and lung cancer tag is close to 1:1. A pneumonia example and a lung cancer example of real EMR text data are shown in Table 1.

5.2 Implementation Details

Our model is implemented in Python with TensorFlow library. For the generator, the word embedding dimension and the hidden state dimension of LSTM cell are both set to 32. The word embedding layer are jointly trained with the generator. For the discriminator, the word embedding dimension is also set to 32. To alleviate the overfitting, we add a dropout layer with 0.2 drop rate before the fully-connected layer and L2 regularization of weights with $\lambda = 0.1$. The word embedding layer is fixed with pretrained weights by Word2Vec [24]. We first pretrain the generative model 1000 epochs by MLE and pretrain the discriminative model 100, 100 epochs for CNN and BiRNN-attention separately by minimizing the cross-entropy between synthetic samples and real samples. In the adversarial training, we update the generator five steps and then update the discriminator five steps. Each REINFORCE step is companied with one step of teacher forcing. The total number of adversarial training epochs is 100. A pneumonia example and a lung cancer example of generated synthetic EMR text data are shown in Table 2. It is worth noting that the real data and synthetic data are both translated automatically with little manual modification.

5.3 Micro-Level Evaluation

We test different discriminator structures and rescale methods with micro-level evaluation metrics in this subsection. To evaluate the model performance, we use the negative log-likelihood on the test set (NLL-test) and self-BLEU [25] as evaluation metrics.

Equation (12) expands NLL-test over the temporal dimension. NLL-test evaluates the model's capacity to fit the real data. The lower NLL-test indicates the higher likelihood for the model to generate samples subject to the similar data distribution to that of the test set, and thus reflects the stronger fitting capacity of the model. The calculation of NLL-test can be formulated as:

$$\mathrm{NLL}_{test} = -\mathbb{E}_{\mathbf{x}\in S_{\mathrm{test}}}\left[\sum_{t=1}^{T}\log G_\theta(x_t|X_{1:t-1})\right]. \tag{12}$$

Self-BLEU is a metric to evaluate the diversity of generated synthetic sentences. BLEU (Bilingual Evaluation Understudy)

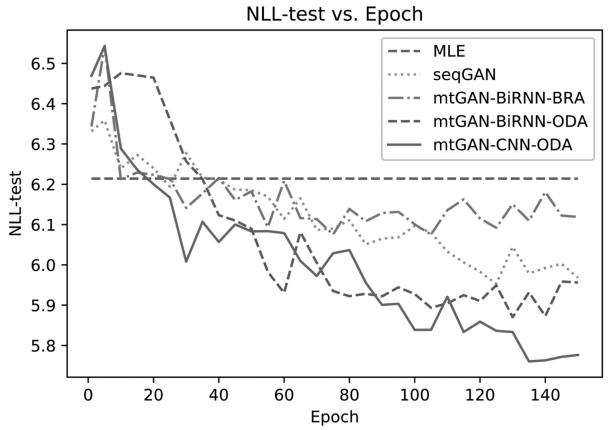


Fig. 2. *NLL-test results*. The figure shows the NLL-test curves with different discriminators and rescale reward methods.

is widely used in the machine translation field and aims to evaluate how similar two sentences are. Inspired by it, we can calculate BLEU between each sentence and the rest in a group of generated sentences, which indicates how this sentence resembles with others. The average BLEU score of each generated sample is defined as self-BLEU, which evaluate the diversity of current generative model. The higher self-BLEU score indicates more commonness among generated samples and thus less diversity. The calculation of self-BLEU can be formulated as:

$$\text{self-BLEU} = \mathbb{E}_{\mathbf{x}\in S_{\text{test}}}[\text{BLEU}(X|\complement_{S_{\text{test}}}(X))]. \tag{13}$$

The NLL-test scores of MLE training and adversarial training with different discriminative models and rescale reward methods are shown in Fig. 2. SeqGAN [26] is an extended conditional model based on the original version and does not use any rescale reward methods, which we also view as a baseline model together with MLE. It can be seen that the rescale reward method ODA is better than BRA and no rescale method. Comparing the curves of MLE, SeqGAN, mtGAN-BiRNN-ODA and mtGAN-CNN-ODA, it is obvious that our mtGAN with CNN discriminator can achieve a better NLL-test score than BiRNN discriminator. In addition, all models trained by adversarial training are better than the model trained by MLE.

The NLL-test and self-BLEU scores are shown in Table 3. We can see that GAN-based approaches outperforms Maximum Log-likelihood Estimation approach in both NLL-test and self-BLEU metrics. Comparing mtGAN-BiRNN-BRA with mtGAN-BiRNN-ODA, we can see that the Optimal Discriminator Activation (ODA) can achieve lower NLL-test at the cost of higher self-BLEU. Since the quality of synthetic EMR text is prior to the diversity, we use ODA to rescale reward. The mtGAN-CNN model can achieve highest NLL-test score and relatively low self-BLEU score, which indicates that our model has the strong ability to fit the real data and generate diverse examples at the same time. We will use mtGAN-CNN-ODA as our main model to conduct the following evaluation.

TABLE 3
Micro-Level Experiment Results

Model	NLL-test	self-BLEU
MLE	6.2141	0.9270
SeqGAN	5.9685	0.9267
mtGAN-BiRNN-BRA	6.1191	**0.9155**
mtGAN-BiRNN-ODA	5.9561	0.9209
mtGAN-CNN-ODA	**5.7764**	0.9182

TABLE 4
Macro-Level Experiment Results

Model	AdverSuc	ERE1	ERE2	ERE3	meanERE
MLE	0.3007	0.0068	0.0676	0.3446	0.1396
SeqGAN	0.1351	0.0203	0.0270	0.1081	0.0518
mtGAN	**0.3041**	0.0811	0.0270	0.2804	0.1295

5.4 Macro-Level Evaluation

For the macro-level evaluation, we conduct an adversarial evaluation experiment similar to [10] to fairly evaluate the whole quality of synthetic EMR texts. The idea of adversarial evaluation resembles the idea of Turing test. In the adversarial evaluation, we train a separate machine evaluator in place of the human evaluator to distinguish real EMR texts and generated synthetic EMR texts. We report Adversarial Success (AdverSuc) on the test set, which is the fraction of generated samples that can fool the evaluator. Higher scores of AdverSuc indicates that the generated synthetic samples are more similar to real samples. However, the adversarial evaluation is model-dependent, a poor discriminative model can also lead to a low accuracy of the evaluator, in which case we can not tell one group of generated examples is definitely better than the other one. Thus, we also set up three manually-designed situations to measure the capacity of the evaluator to correctly distinguish real EMR texts and synthetic EMR texts.

1) Randomly split real EMR texts as positive examples and negative examples. An ideal evaluator should give an accuracy of 0.5.
2) Randomly split generated EMR texts as positive examples and negative examples. An ideal evaluator should also give an accuracy of 0.5.
3) Use real EMR texts as positive examples and random generated EMR texts as negative examples. An ideal evaluator should give an accuracy of 1.0.

We report the absolute value of the difference between the evaluator's accuracy and the ideal accuracy in above three experiments as evaluator reliability error (ERE). The lower ERE value indicates the model is closer to the ideal model and thus have the higher model reliability. We train a separate discriminator in the adversarial evaluation experiment. The AdverSuc and ERE scores of MLE, SeqGAN and our mtGAN are shown in Table 4. Our mtGAN model can achieve the highest AdverSuc and lower meanERE than MLE at the same time, which indicates the synthetic EMR texts generated by our model have the higher quality than samples generated by MLE, and this result is reliable. It is hard to compare the results between SeqGAN and mtGAN, as we are not sure what mtGAN's AdverSuc will be if the evaluator is more reliable. However, from Table 3, we can see that SeqGAN, which does not use any method to rescale

TABLE 5
Application-Level Experiment Results

Accuracy / Data source / Model	Real	Synthetic	Mix
MLE	0.7500	0.7432	0.7568
SeqGAN	0.7500	0.6959	0.7095
mtGAN	0.7500	0.7432	**0.7635**

reward, can not be trained efficiently and has a low capcity to fit the data. It tends to generate more repetitive synthetic samples. Generally, the evaluator can obtain less noisy data to learn what is real EMR text and thus the ERE score is lower than other methods. The AdverSuc of SeqGAN is pretty low, which implies the quality of synthetic EMR texts generated by SeqGAN is not satisfactory.

5.5 Application-Level Evaluation

For the application-level evaluation, we design a practical classification experiment to test if the samples generated by our model can be used as in real-world scenarios, as the source of data augmentation and help other machine learning tasks. In this classification experiment, the labels are pneumonia and lung cancer, which is set same to the conditional constraint used in the generative model training. It is worth noticing that during the whole training process (from pretraining to adversarial training), our model does not get access to the test set, so we can assign labels to our model to generate corresponding samples, and use these synthetic EMR texts to train a classifier and evaluate it on the test set. Ideally, using generated samples to train a classifier should achieve similar performance to using real samples, and adding generated samples to real samples should achieve a higher score as the result of data augmentation.

The application-level evaluation results are shown in Table 5. It can be seen that with only synthetic examples to train a classifier, the classifier shows similar performance with examples generated by our model and MLE, and it is better than SeqGAN. Next, we train a model with a mixed dataset, where synthetic EMR examples are added to real EMR examples to construct a larger dataset. First, we can see that the classifier is better than the model trained with only real examples or synthetic examples, which indicates synthetic EMR texts can indeed be used as an approach of data augmentation. Additionally, it can be seen that the synthetic EMR texts generated by our model has a better quality compared with examples generated by MLE, so the classifier shows higher classification ability. Our approach to generate synthetic EMR texts can be used as a data augmentation approach to assist other tasks like the disease classification task. In addition, our proposed approach can generally serve as a data source to develop other machine learning algorithms in the medical field, where the privacy-sensitive EMR text is scarce.

5.6 Example Analysis and Discussion

Finally, we will analyze some examples generated by our model and discuss the advantage and disadvantage of our model. Furthermore, we will discuss the promising and challenging parts of synthetic EMR text generation.

In Tables 6 and 7, we show several *good* examples and *bad* examples generated by our model, which are chosen by subjective judgements of people. For those *good* examples, we can find some common features of them: words matching, grammar correct and logical. They are read well by humans and sometimes we even could not distinguish them from real EMR texts. For those *bad* examples, there are some typical problems of them, such as repetitive, inconsistent and improper words matching. Parts of sentences that have problems are marked as bold text in the Table 7. For example, in the first example of type *Inconsistent*, there is a description in the synthetic EMR text that the patient has a highest body temperature of 39.5 °C. However, the next sentence says the patient doesn't have a fever, which is obviously contradictory with the previous description. These examples indicate that the consistency and the logic of the whole text are still the biggest challenges of the long EMR text generation.

There are several possible approaches to solve or alleviate these problems to an extent. Considering rewards play the most important role in the adversarial learning, one way is to combine rewards given by discriminator with human-designed rewards, so that the rewards can give more precise guiding signals to the model training. Another way is to apply constraints on the generation phase to improve the readability of synthetic EMR texts. It is worth noticing that our model is based on the GAN framework, while the other branch of generative models variational auto-encoder [27] may also apply to the EMR text generation problem. We believe such a encoder-decoder model is also a promising method as it can build a map between a noise distribution

TABLE 6
Good Generated Examples That are Words Matching, Grammar Correct, and Logical

Examples
1. 患者于1周前无明显诱因出现咳嗽，咳白色粘痰，伴活动后加重，休息后可缓解，间断服用镇咳药物等治疗，未行正规诊治。 (The patient had no obvious cause of coughing 1 week ago, coughing white sticky, increased after exercise, relieved after rest, intermittent treatment with antitussive drugs, etc., no formal diagnosis and treatment.)
2. 患者于3天前无明显诱因出现咳嗽、咳痰，为黄色粘痰，伴活动后气喘，一年四季均有发作，无气喘，无咯血，无咽痛，无胸痛，有时伴发热 (The patient had no obvious cause of coughing and coughing 3 days ago. It was yellow sticky, accompanied by asthma after exercise. It had seizures all year round, no asthma, no hemoptysis, no sore throat, no chest pain, sometimes accompanied by fever.)
3. 患者无明显诱因出现反复出现腹胀、腹泻，稀水样便，有恶心、呕吐，呕吐物为胃内容物。无畏寒、发热。就诊于当地社区医院，诊断为"急性肠炎" (The patient had no obvious cause of repeated abdominal distension, diarrhea, dilute watery stool, nausea and vomiting, and vomit was the stomach contents. No chills, fever. Visiting a local community hospital, diagnosed as "acute enteritis")
4. 患者2014.6查颈部不适。无明显诱因出现轻微咳嗽，以干咳为主，咳痰，伴气喘，无发热，于当地医院给予左氧氟沙星抗感染治疗未见腺癌 (Patient 2014.6 checked neck discomfort. There was no obvious cause of mild cough, mainly dry cough, cough, asthma, no fever. Levofloxacin anti-infective treatment was not treated in local hospital.)
5. 患者10多年前开始出现咳嗽、咳痰，痰中带血，当地医院查胸部CT示纵隔肿大淋巴结，右肺下叶结节，后于北京人民医院行气管镜时治疗收入院。 (The patient began to have cough and cough more than 10 years ago, and blood in the sputum. The local hospital examined chest CT to show mediastinal lymph nodes, right lower lobe nodules, and then treated the hospital at the Beijing People's Hospital for bronchoscopy.)

TABLE 7
Bad Generated Examples That are Repetitive, Inconsistent, and Improper Word Matching

Type	Examples
Repetitive	1. 患者9年前受凉后出现咳嗽、咳痰、气喘，活动后加重，喘憋明显，**给予抗感染等对症治疗后好转** (Improved after giving anti-infection symptomatic treatment)，后进食后症状可改善，**给予抗感染等对症治疗后好转** (Improved after giving anti-infection symptomatic treatment) 2. 患者院外病情尚稳定，10年前出现**无发热** (no fever)，咳嗽、咳痰，痰为白色粘痰，伴咯血、气喘、憋气，尚可平卧，**无发热** (no fever)，无畏寒、寒战，体温最高 3. 患者1天前无明显诱因出现咳嗽、较剧烈，**伴咳少许白色粘痰** (with coughing a little white sticky sputum)，间断出现咳嗽，**咳白色痰，粘稠** (cough white sputum, sticky)，不易咳出，感胸闷减轻，伴气喘，不伴有心前区
Inconsistent	1. 患者无明显诱因出现咳嗽、咳痰渐感咳嗽、咳白色粘痰，**最高体温39.5°C** (Maximum body temperature 39.5°C)，**无**咳血(no)、**发热** (fever)，无畏寒 2. 患者2014.1无明显诱因出现咳嗽，**以干咳为主，无痰** (Mainly dry cough, no sputum)，**无咳嗽、咳痰** (no cough, no sputum)，无咯血、胸痛，无发热，无胸痛、胸闷、心悸，无畏寒
Improper word matching	1. 患者反复出现反复咳嗽，咳白色粘痰，伴活动后气短，无午后发热，体温痰量增多、**体温阵发性咳嗽** (body temperature paroxysmal cough)，就诊于北京东直门医院，镜身不能慢性阻塞性肺疾病合并肺部感染 2. 患者于3年前无明显诱因出现咳嗽、咳痰，每年发病后出现气喘，出现颜面部及眼睑呕血憋气。**呼吸无欠佳后体温** (breathe no abnormal body temperature)就诊于当地医院查发现右上叶及肺囊肿切除术

and an EMR text distribution, so that it can provide a more convenient approach for the low-dimensional representation of EMRs, the similarity evaluation between EMRs, which can help us have a better understanding of diseases.

6 Conclusion

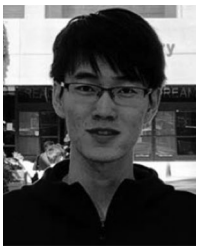
In this paper, we propose a conditional model mtGAN to generate synthetic EMR texts. This method solves the privacy problem and the insufficient and imbalance samples problem naturally. The micro-level, macro-level and application-level evaluation demonstrate that our model can generate more realistic EMR texts, which has a wide range of applications. The future work will focus on the hidden representation of EMR texts, which will help us impose more direct control to the generation process and also help us analyze the similarity of different EMR texts to better understand diseases.

Acknowledgments

This work is supported in part by the National Key R&D Program of China (Grants 2018YFC0910401 and 2018YFC0910404) and and National Natural Science Foundation of China (Grants 61721003 and 11801301).

References

[1] K. Wang, R. Chen, B. C. Fung, and P. S. Yu, "Privacy-preserving data publishing: A survey on recent developments," *ACM Comput. Surveys*, vol. 42, 2010, Art. no. 1.
[2] K. El Emam, E. Jonker, L. Arbuckle, and B. Malin, "A systematic review of re-identification attacks on health data," *PloS One*, vol. 6, no. 12, 2011, Art. no. e28071.
[3] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*, Berlin, Germany: Springer, 2012.
[4] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Proc. Advances Neural Inf. Process. Syst.*, 2015, pp. 1171–1179.
[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Advances Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.
[6] L. Yu, W. Zhang, J. Wang, and Y. Yu, "SeqGAN: Sequence generative adversarial nets with policy gradient," in *Proc. 31st AAAI Conf. Artif. Intell.*, Mar. 2017, pp. 2852–2858.
[7] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. (3–4), pp. 229–256, 1992.
[8] S. Hochreiter, and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[9] K. Cho *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," 2014, *arXiv preprint arXiv:1406.1078*.
[10] J. Li, W. Monroe, T. Shi, S. Jean, A. Ritter, and D. Jurafsky, "Adversarial learning for neural dialogue generation," 2017, *arXiv preprint arXiv:1701.06547*.
[11] R. J. Williams, and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, vol. 1, no. 2, pp. 270–280, 1989.
[12] E. Choi, S. Biswal, B. Malin, J. Duke, W. F. Stewart, and J. Sun, "Generating multi-label discrete patient records using generative adversarial networks," in *Proc. Mach. Learn. Healthcare Conf.*, Nov. 2017, pp. 286–305.
[13] C. Esteban, S. L. Hyland, and G. Rätsch, "Real-valued (medical) time series generation with recurrent conditional GANs," 2017, *arXiv preprint arXiv:1706.02633*.
[14] A. Yahi, R. Vanguri, N. Elhadad, and N. P. Tatonetti, "Generative adversarial networks for electronic health records: A framework for exploring and evaluating methods for predicting drug-induced laboratory test trajectories," 2017, *arXiv preprint arXiv:1712.00164*.
[15] C. B. Browne *et al.*, "A survey of monte carlo tree search methods," *IEEE Trans. Comput. Intell. Ai Games*, vol. 4, no. 1, pp. 1–43, Mar. 2012.
[16] M. Mirza, and S. Osindero, "Conditional generative adversarial nets," *Comput. Sci.*, 2014, *arXiv preprint arXiv:1411.1784*.
[17] Y. Kim, "Convolutional neural networks for sentence classification," 2014, *arXiv preprint arXiv:1408.5882*.
[18] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *Comput. Sci.*, 2014.
[19] S. Lu, Y. Zhu, W. Zhang, J. Wang, and Y. Yu, "Neural text generation: Past, present and beyond," 2018, *arXiv preprint arXiv:1803.07133*.
[20] T. Che *et al.*, "Maximum-likelihood augmented discrete generative adversarial networks," 2017, *arXiv preprint arXiv:1702.07983*.
[21] J. Guo, S. Lu, H. Cai, W. Zhang, Y. Yu, and J. Wang, "Long text generation via adversarial training with leaked information," in *Thirty-Second AAAI Conf. Artif. Intell.*, Apr. 2018.
[22] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, 2015, Art. no. 529.
[23] Jieba Chinese Text Segmentation, 2013, Accessed: Feb. 1, 2018. [Online]. Available: https://github.com/fxsjy/jieba
[24] Google Word2Vec, 2013, Accessed: Feb. 1, 2018. [Online]. Available: https://code.google.com/archive/p/word2vec
[25] Y. Zhu, S. Lu, L. Zheng, J. Guo, W. Zhang, and J. Wang *et al.*, "Texygen: A benchmarking platform for text generation models," 2018, *arXiv:1802.01886*.
[26] Implementation of Sequence Generative Adversarial Nets with Policy Gradient, 2017, Accessed: Mar. 1, 2018. [Online]. Available: https://github.com/LantaoYu/SeqGAN
[27] D. P. Kingma, and M. Welling, "Auto-encoding variational bayes," 2013, *arXiv:1312.6114*.



Jiaqi Guan received the BA degree from the Department of Automation, Tsinghua University. He is currently working toward the PhD degree in the Department of Computer Science, Univeristy of Illinois Urbana-Champaign. His research interests include machine learning and generative models.

Runzhe Li received the BS degree from the Department of Mathematical Sciences, Tsinghua University. He is currently working toward the PhD degree in the Department of Biostatistics, Johns Hopkins University. His current research interests include developing statistical methodologies and machine learning (deep learning) approaches to address scientific problems in single-cell genomics, electronic medical records, etc.

Sheng Yu received the BS and MA degrees in statistics from Nankai University and the University of Michigan, Ann Arbor, and the PhD degree in systems engineering from George Washington University. From 2012 to 2015, he was research fellow at Harvard Medical School and Partners HealthCare Personalized Medicine. He is currently associate professor of the Center for Statistical Science and the Department of Industrial Engineering, and RONG professor of the Institute of Data Science of Tsinghua University. His major research topics are electronic health records data analysis and automated construction of large scale medical knowledge graphs.

Xuegong Zhang received the BA and PhD degrees from the Department of Automation, Tsinghua University. He is currently professor of Department of Automation in Pattern Recognition and Bioinformatics, director of Bioinformatics division, BNRIST, and deputy director of MOE Key Laboratory of Bioinformatics. His major research topics are machine Learning, pattern recognition, bioinformatics, computational genomics and systems biology. He is a member of the IEEE.