Automatic generation of sentimental texts via mixture adversarial networks

K. Wang a, b, X. Wan a, b, *

a Institute of Computer Science and Technology, Peking University, China
b The MOE Key Laboratory of Computational Linguistics, Peking University, China

A R T I C L E   I N F O

Article history:
Received 18 October 2018
Received in revised form 29 March 2019
Accepted 5 July 2019
Available online 19 July 2019

Keywords:
Natural Language Generation
Sentimental Text Generation
Generative Adversarial Net
SentiGAN
C-SentiGAN

A B S T R A C T

Automatic generation of texts with different sentiment labels has wide use in artificial intelligence applications such as conversational agents. It is an important problem to be addressed for achieving emotional intelligence. In this paper, we propose two novel models, SentiGAN and C-SentiGAN, which have multiple generators and one multi-class discriminator, to address this problem. In our models, multiple generators are trained simultaneously, aiming at generating texts of different sentiment labels without supervision. We propose a penalty-based objective in generators to force each of them to generate diversified examples of a specific sentiment label. Moreover, the use of multiple generators and one multi-class discriminator can make each generator focus on generating its own texts of a specific sentiment label accurately. Experimental results on a variety of datasets demonstrate that our SentiGAN model consistently outperforms several state-of-the-art text generation models in the sentiment accuracy and quality of generated texts. In addition, experiments on conditional text generation tasks show that our C-SentiGAN model has good prospects for specific text generation tasks.

© 2019 Elsevier B.V. All rights reserved.

1. Introduction

Emotional intelligence is an important part of artificial intelligence [1–7]. Automatic understanding and generation of sentimental texts not only make machines more friendly to humans, but also make them look more intelligent, especially in some computer applications that interact with people, such as dialogue systems or conversational agents [8, 9], automated commenting and feedback systems [10–13]. Fig. 1 shows several example responses in a dialogue with/without considering sentiment. The red block is a response that does not consider sentiment. The green block contains responses with various emotion types (from top to bottom: happiness, love, sadness, disgust and anger), which seem to be more intelligent and friendly to human users.

Nowadays, with the use of deep learning [14], sentiment classification on short texts has made good progress. For instance, one of the state-of-the-art sentiment classifiers has achieved an accuracy of 90% on the Stanford Sentiment Treebank dataset [15]. But compared with the success of sentiment classification, generic sentimental text generation is challenging and very few recent attempts have been made to investigate it. Previous work has been mostly limited to task-specific

* This paper is an invited revision of a paper which first appeared at the 2018 International Joint Conference on Artificial Intelligence (IJCAI-18).
* Corresponding author at: Institute of Computer Science and Technology & The MOE Key Laboratory of Computational Linguistics, Peking University, China.
E-mail addresses: wangke17@pku.edu.cn (K. Wang), wanxiaojun@pku.edu.cn (X. Wan).
https://doi.org/10.1016/j.artint.2019.07.003
0004-3702/© 2019 Elsevier B.V. All rights reserved.
applications and just uses hidden variables to indirectly control the sentiment labels of generated texts, especially in emotional response generation [16,17]. An important reason is that it is difficult to design an appropriate and specific training objective in deep generative models for sentimental text generation. Inspired by Generative Adversarial Net (GAN) [18], which uses a discriminator instead of a specific objective to guide the generator, our motivation is two-fold: (1) Since text sentiment classification is very strong, we can use the classifier to guide the generation of sentimental texts. (2) Due to the difficulty of designing specific sentiment-related training objective, we use a discriminator network to learn the loss function of sentiment-related goal over time, rather than explicitly formulating it.

In this study, we hope to use GAN to control sentimental text generation, and aim to generate texts with higher quality, more variety and more accurate sentiment polarity. However, there are a few challenges to be addressed when applying GAN to generate sentimental texts. Firstly, the discrete nature of texts leads to a sampling step that is not differentiable, which makes it impossible for the gradient to pass from the discriminator to the generator. Recently, some studies use reinforcement learning which treats the process of discriminator guiding generator as a reinforcement learning policy [19, 20]. Nonetheless, the generated texts face the problem of poor quality. Secondly, one of the major drawbacks of GAN is the problem of ‘mode collapse’, and it has been empirically proven that GAN prefers to generate samples around only a few modes whilst ignoring other modes [21] (as shown in Fig. 2(a)). So there is a lack of diversity in generated texts.

To address the above issues, we propose two text generation models, SentiGAN and C-SentiGAN, for generic and conditional sentimental text generation, respectively. Our models consist of multiple generators and one multi-class discriminator, which are trained simultaneously. Like [19], we consider the sequence generation procedure as a sequential decision making process. That is, we regard each generation model as a stochastically parametrized policy and use Monte Carlo search to approximate the state-action value. Then we use the discriminator to evaluate the sequence and guide the learning of the generation model. But unlike previous works, our models contain multiple generators and one discriminator. First, we propose a novel penalty-based objective, which adopts a more reasonable measure and aims to minimize overall penalties instead of maximizing rewards (as shown in Fig. 2(b)). It is proved both experimentally and theoretically that, our penalty-based objective can force each generator to generate diversified texts of a specific sentiment label, rather than generating examples which are repetitive but ‘safe’ and ‘good’. Second, the use of our discriminator’s multi-class classification objective can make generators more focused on generating their own examples of specific sentiment labels, and stay away from other types of sentiments (as shown in Fig. 2(c)). This improves the sentiment accuracy of the generated texts. Third, our C-SentiGAN has an additional encoder that encodes conditional text, allowing our model to handle conditional input text, which is more suitable for specific real-world text generation tasks, such as dialog systems.

For generic text generation, we compare our SentiGAN with several state-of-the-art deep generative models, including RNNLM [22], GAN and its variants [19,20,23–25], VAEs [26,27]. For conditional text generation, we compare our C-SentiGAN with Seq2Seq model [28,29], Conditional GAN (C-GAN) [30], Conditional Variational Autoencoder (C-VAE) [31]. We take a closer look at the generation quality as well as our models’ capability of expressing sentiment. First, we evaluate the generated text from two aspects: the first is the sentiment accuracy of the generated texts, which is achieved by a well-performed sentiment classifier and manual evaluation; the second is the quality of the generated texts, which is achieved by several metrics (i.e., fluency, novelty, diversity, intelligibility) designed by us. Second, we analyze our models more deeply, including training curves, model complexity, sensitivity of metrics, influence of the number of sentiment type, and exploration of descriptive text generation. Experimental results on various datasets (i.e., movie reviews, beer reviews, custom reviews, a
descriptive text corpus, twitter emotional conversations, and a synthetic dataset) demonstrate that our models consistently outperform baseline models in both the sentiment accuracy and quality of generated texts. Moreover, the superiority of our C-SentiGAN shows good prospects for applying to other real-world specific text generation tasks.

Part of the content of our first model - SentiGAN has been published in [32], the extensions made in this article are summarized as follows: (1) We propose a new extended model, C-SentiGAN, to deal with the specific real-world sentimental text generation problem that has conditional input text. (2) We add more details and analysis of our models and experiments, e.g., theoretical analysis, background and related work, model complexity, model parameters, data processing, training curves, sensitivity of metrics, and influence of the number of sentiment types. (3) We compare our models with more baselines and conduct more experiments to evaluate our models’ capability, such as exploration of descriptive text generation and emotional conversation generation.

The major contributions of this study are summarized as follows:

1. We propose a novel framework, SentiGAN, to generate generic, diversified and high-quality sentimental texts of different sentiment labels.
2. We propose an extended model, C-SentiGAN, to deal with the conditional sentimental text generation problem.
3. We propose a new penalty-based objective to make each generator in our models produce diversified texts of a specific sentiment label.
4. Extensive experiments are performed on a variety of datasets and the results demonstrate the efficacy and superiority of our proposed models.

In the rest of this paper, we first introduce related work in Section 2. The framework and technical details of our proposed models are described in Section 3. In Section 4, we present extensive experimental results to demonstrate the efficacy of our proposed models. Lastly, we conclude this paper and point out future directions in Section 5.

2. Related work

2.1. Unsupervised text generation

Unsupervised text generation is an important research area in natural language processing [28,33,34]. A standard recurrent neural network (RNN) language model [22] predicts each word of a sentence conditioned on the previous word and an evolving hidden state. However, it suffers from two major drawbacks when used to generate texts. First, RNN based models are always trained through the maximum likelihood approach, which suffers from exposure bias [35]. Second, the loss function used to train the model is at the word level but the performance is typically evaluated at the sentence level. Later, Professor Forcing [36] is proposed to train RNN by using a discriminator to discriminate the hidden states of a generator RNN that is conditioned on real and synthetic samples. Scheduled Sampling [35] and MaskGAN [37] are proposed to randomly adjust the sampling words during training time. But those approaches do not directly specify the loss function on the RNN output to ensure high sample quality. There are some researches which use the generative adversarial network (GAN) to solve the problems.

2.2. Generative adversarial nets

Generative Adversarial Nets (GANs) [18] are a recent novel class of deep generative models. Though GANs achieve great successes on computer vision applications [38–46,40,47,48], there are only a few progresses in natural language generation because the discrete sequences are not differentiable. Some works attempt to solve this problem, including Gumbel-softmax distribution (GSGAN) [25], Professor Forcing [36], MaliGAN [24] and so on. However, it is more common to tackle the non-differentiable problem with a strategic gradient of reinforcement learning, including SeqGAN [19], RankGAN [23], LeakGAN [20]. The effects of these variants of GANs are not very different, and none of these methods can generate samples with diverse attributes. Without loss of generality, we will focus on comparison with SeqGAN in this study. Conditional GAN [30] is a variant of GAN that produces controlled samples by using a condition variable to guide the generation. This is also one of our main comparisons. LabelGAN [48] uses a discriminator to identify multiple categories and is similar to us, but it has only one generator and does not solve discrete problems in text generation. Other superior unsupervised deep generative models include Variational Autoencoders (VAE) [26], semi-supervised VAE (S-VAE) [27]. VAE consists of encoder and generator networks which encode a data example to a latent representation and generate samples from the latent space, respectively. Although VAE does not have the problem of generating discrete data, it has more constraints and restrictions than GAN, and we will also compare our model with it in the experiments.

2.3. Sentimental text generation

Sentimental text generation is attracting the attention of researchers and is increasingly being needed in some artificial intelligence systems, such as dialogue system or conversational agents [16,17,8,9,49], automated commenting and feedback systems [10–13]. This is because it not only makes machines more human-friendly, but also makes them look smarter.
However, previous works have been mostly limited to task-specific applications and just use hidden vectors to indirectly control the generation of sentiment [16,17], and the reason is that it is difficult to design an appropriate, specific training objective for sentiment generation. Conditional Variational Autoencoder (C-VAE) [31] model is proposed to generate specific sentimental texts by incorporating conditional option in the generative process. Unlike them, our models directly consider sentiment as the target and generate sentimental texts via policy gradient. Moreover, our model interacts with multiple generators to further improve the sentiment accuracy and the quality of the generated sentences.

2.4. Conditional text generation

Conditional text generation aims to generate texts conditioned on specific inputs, such as dialogue system [16,17,8,9], machine translation [50], product review generation [51,52,12]. These methods usually need a large parallel corpus for learning an encoder-decoder and use encoder network to encode conditional input text. The most commonly used model is Seq2Seq model [28,29], which uses two RNNs to build the encoder-decoder framework with global attention mechanism [53]. However, due to the deficit of parallel data with the same conditional inputs and different-style responses, the sentiment of the output of such models is generally consistent with the majority of the training corpus and cannot be artificially controlled. Different from these studies, we aim to generate a variety of generic texts (and conditional responses) of different sentiment labels.

3. Method

In this section, we first analyze the background knowledge of current text generation models, including the maximum likelihood estimation model and generative adversarial nets. And then we introduce our models (i.e., SentiGAN and C-SentiGAN) in detail. Finally, we theoretically analyze the multi-class classification objective and the penalty-based objective we propose.

3.1. Background

3.1.1. Maximum likelihood estimation approach

Nowadays, the dominant approach to text generation is based on the maximum likelihood estimation (MLE) model. MLE is often used for estimating the parameters of a statistical text generation model by giving the observations, and then maximizing the corresponding likelihood function. More specifically, given the generator $G$ and a training sentence $x = \{x_0, x_1, \ldots, x_t, \ldots, x_{|x|-1}\}$ with length $|x|$, MLE approach aims to maximize the likelihood function:

$$\mathbb{E}\left[ \prod_{t=1}^{|x|-1} G(x_t|x_{0:t-1}; \theta_g) \right],$$

where $\theta_g$ is the parameter of the autoregressive generation model $G$, which can be a recurrent neural network [54] or its variants (e.g., LSTM [55], GRU [56]). During the period of training, generator $G$ receives the ground truth $x_{0:t-1}$ as input at time step $t$, and the objective of generator is to minimize the negative log-likelihood loss as follows:

$$L_{\text{MLE}}(G(z; \theta_g), x; \theta_g) = \mathbb{E}[- \sum_{t=1}^{|x|-1} \log G(x_t|x_{0:t-1}; \theta_g)].$$

where $G(x_t|x_{0:t-1}; \theta_g)$ denotes the probability that the output of $G$ is $x_t$ under the condition of the former given sequence $x_{0:t-1} = \{x_0, x_1, \ldots, x_{t-1}\}$ at time step $t$. While in the inference phase, generator $G$ will take the previous sampled output $y_{0:t-1}$ as the input at time step $t$.

Fig. 3 depicts the different phases of the MLE approach. However, the maximum likelihood approach may be faced with two significant problems. First, the standard MLE training only considers the word-level objective, which is prone to local optimization and difficult to guarantee sentence-level goals (e.g., sentiment polarity, preserving semantic consistencies, etc.). Second, the model may be affected by exposure bias [35] due to the dependence on the previously sampled output during the inferring phase.
3.2.2. Generative adversarial nets

Generative adversarial networks (GAN) [18] are widely used in the computer vision (CV) field [38–46], which are mainly used to ensure a higher-level label resembling qualities such as realism [38–42,45], style [43,42], smoothness [38,46], preserving neighborhood consistencies [46], and so on. This is done by using a discriminator network that learns the loss function of these expected attributes over time, rather than explicitly formulating these attributes. A typical generative adversarial network is a generator and a discriminator play a min-maximum game, and the discriminator is trained to distinguish real from fake data, while the generator is trained to generate fake data that the discriminator will mistakenly recognize as real. The objective function of a standard GAN is shown below:

\[
\min_{G} \max_{D} J_{\text{gan}}(\theta_{G}, \theta_{D}) = \mathbb{E}_{x \sim P_{r}}[\log D(x)] - \mathbb{E}_{x \sim P_{g}}[\log(1 - D(x))],
\]

where \(P_{r}\) and \(P_{g}\) are the distributions of real data and data generated by the generator, respectively. However, the discreteness of texts leads to a sampling step that is not differentiable, which makes it difficult for the gradient to pass from the discriminator to the generator. An intuitive choice is to use some differentiable sampling functions, such as gumbel-softmax [25], but this makes the model less stable and the effect of back-propagation is limited. Another more common alternative is to use reinforcement learning which treats the process of discriminator guiding generator as a reinforcement learning policy [19,20,23,32]. However, due to the more complex language patterns and the mode collapse problem of GAN, the quality of generated text is usually not very satisfactory.

3.2. SentiGAN & C-SentiGAN

3.2.1. SentiGAN

The overall framework of our proposed SentiGAN is shown in Fig. 4. Supposing we want to generate texts with \(k\) types of sentiments (i.e., \(k\) sentiment labels), we use \(k\) generators \(G_{i}(x_{t}|x_{t-1}; \theta_{g}^{i})\) and one discriminator \(D(x; \theta_{d})\), where \(\theta_{g}^{i}\) and \(\theta_{d}\) are the parameters of the \(i\)-th generator and the discriminator respectively, and the prior input noise \(z\) sampled from the distribution \(P_{z}\) (e.g., a normal distribution) is used to initialize the generator’s input.

The whole framework can be divided into two adversarial learning objectives: generator learning and discriminator learning. The goal of the \(i\)-th generator \(G_{i}\) is to generate texts with the \(i\)-th sentiment type that can deceive the discriminator. Specifically, it aims to minimize the penalty-based objective that we propose. In contrast, the goal of the discriminator is to distinguish between fake texts (texts generated by generators) and real texts with \(k\) sentiment types as much as possible, which is the multi-class classification objective we adopt.

3.2.2. C-SentiGAN

As we mentioned earlier, SentiGAN is a generic sentimental text generation model that uses sampled random noise as initial input \(z\) of the generator. Further, in order to apply SentiGAN to conditional text generation tasks, we propose C-SentiGAN, which is better suited for migrating to other specific text generation tasks with conditional input text, such as dialog systems. Fig. 5 shows the framework of C-SentiGAN.

Compared with SentiGAN, our C-SentiGAN adds an additional encoder \(E\) for encoding conditional text and uses the high-level representation of the conditional text instead of noise as the initial input \(z\) of the generator. We add a global attention mechanism [53] between the encoder and generators. The whole model looks like an encoder-decoder framework with global attention mechanism [53] and the decoder part is the same as the SentiGAN and uses the discriminator network as a loss function to ensure high-level objectives, such as sentiment polarity and realism.

3.2.3. Generator learning

To solve the problem that the gradient cannot pass back to the generative model when the output is discrete, we formalize the text generation problem as a sequential decision making process [57]. That is, at each timestep \(t\), we train a generator \(G_{i}\) to produce a sequence \(x_{0:t} = [x_{0}, x_{1}, \ldots, x_{t}]\), where \(x_{t}\) represents a word token in the given vocabulary \(C\).
Given $G_i(\mathbf{x}_t|\mathbf{x}_{0:t-1}; \theta^i_g)$ means the probability of selecting the $t$-th word given the previously generated words (its current state) $\mathbf{x}_{0:t-1}$. And we define a new penalty-based loss function $L(\mathbf{x})$:

$$L(\mathbf{x}) = \sum_{t=1}^{\left|\mathbf{x}\right|-1} G_i(x_t|\mathbf{x}_{0:t-1}; \theta^i_g) \cdot V^{G_i}_{D_i}(\mathbf{x}_{0:t-1} \oplus x_t)$$  \hspace{1cm} (4)$$

where $V^{G_i}_{D_i}(\mathbf{x}_{0:t-1} \oplus x_t)$ is the penalty for sequence $\mathbf{x}_{0:t-1} \oplus x_t$ calculated by the discriminator. $\oplus$ is a concatenation operator, and $\mathbf{x}_{0:t-1} \oplus x_t$ is denoted as:

$$\mathbf{x}_{0:t-1} \oplus x_t = \{x_0, \ldots, x_{t-1}, x_t\} = \mathbf{x}_{0:t}.$$  \hspace{1cm} (5)

However, since the discriminator can only judge on a complete sequence, we apply the Monte Carlo search [58] with roll-out policy $G_i$ to sample the unknown last $|\mathbf{x}| - t$ tokens. We represent an N-time Monte Carlo search as:

$$MC^{G_i}(\mathbf{x}_{0:t}; N) = \{\mathbf{x}_{t+1:|\mathbf{x}|}^1; \ldots; \mathbf{x}_{t+1:|\mathbf{x}|}^N\},$$  \hspace{1cm} (6)

where $\mathbf{x}_{t+1:|\mathbf{x}|}^n$ is sampled based on the roll-out [59] policy $G_i$ and the current state $\mathbf{x}_{0:t}$. That is, we run the roll-out policy $G_i$ starting from current state $\mathbf{x}_{0:t}$ till the end of the sequence for $N$ times to get a batch of output samples $\{\mathbf{x}_{0:t} \oplus \mathbf{x}_{t+1:|\mathbf{x}|}^n\}_{n=1}^N$.

As a consequence, our penalty function for the $i$-th generator is calculated as:

$$V^{G_i}_{D_i}(\mathbf{x}_{0:t-1} \oplus x_t) = \left\{ \begin{array}{ll} \frac{1}{N} \sum_{n=1}^{N} \{1 - D_i(\mathbf{x}_{0:t} \oplus \mathbf{x}_{t+1:|\mathbf{x}|}^n; \theta_d)\} & t < |\mathbf{x}| \\
1 - D_i(\mathbf{x}_{0:t}; \theta_d) & t = |\mathbf{x}| \end{array} \right.$$  \hspace{1cm} (7)

where $D_i(\mathbf{x}_{0:t} \oplus \mathbf{x}_{t+1:|\mathbf{x}|}; \theta_d)$ is the sentence probability given by the discriminator that $\mathbf{x}_{0:t} \oplus \mathbf{x}_{t+1:|\mathbf{x}|}$ is the real text of the $i$-th sentiment type.

Finally, the objective of the $i$-th generator $G_i(\mathbf{x}_t|\mathbf{x}_{t-1}; \theta^i_g)$ is to minimize the total penalty-based value:

$$J_{G_i}(\theta^i_g) = \mathbb{E}_{x \sim P_{\mathbf{x}}}[L(X)] = \mathbb{E}_{x \sim P_{\mathbf{x}}}[\sum_{t=1}^{|x|-1} G_i(x_t|\mathbf{x}_{0:t-1}; \theta^i_g) \cdot V^{G_i}_{D_i}(\mathbf{x}_{0:t-1} \oplus x_t)],$$  \hspace{1cm} (8)

where $\mathbf{x}_t \in C$.

In addition, our generator here is a single layer of Long Short-Term Memory (LSTM) [55] which outputs the $t$-th word according to the distribution:

$$p_t = \text{softmax}(LSTM_{\theta_g}(h_{t-1}, x_{t-1})), \sum_{t \in C} p_t(x_t) = 1$$  \hspace{1cm} (9)

where $\theta_g$ is the parameters of the generator $LSTM_{\theta_g}$, and $h_t$ is the hidden state of timestep $t$. It is worth noticing that our generator can be easily extended to other types of generators as well, such as GRU [60], SRU [61] and Nested LSTM [62].

### 3.2.4 Discriminator learning

Inspired by the discriminator formulation for semi-supervised learning [48], we use a multi-class classification objective that requires the discriminator to distinguish between the real texts of each sentiment type and the generated texts. In more detail, given the set of $k$ generators, the discriminator produces a softmax probability distribution over $k+1$ classes. The score at $i$-th ($i \in \{1, \ldots, k\}$) index ($D_i$) represents the probability that it belongs to the real texts of the $i$-th sentiment type, and the score at ($k+1$)-th index represents the probability that the sample is generated by generators. The objective function of the discriminator is to minimize:
\[ J_D (\theta_d) = -\mathbb{E}_{x \sim P_g} \log D_{k+1}(x; \theta_d) - \sum_{i=1}^{k} \mathbb{E}_{x \sim P_i} \log D_i(x; \theta_d) \]  

(10)

where \( P_g \) is texts generated by all generators, \( P_i \) is real texts of the \( i \)-th sentiment type, and \( D_i(x; \theta_d) \) represents the score at the \( i \)-th index of \( D(x; \theta_d) \). Since CNN has recently been shown of great effectiveness in text classification \([63]\), our discriminator here is a layer of CNN which has multiple filters. More specifically, we first represent each word \( x_t \) as a fixed-size vector \( w_t \in \mathbb{R}^d \) (\( d \) is the dimension of the word vector) by looking up from word embeddings, so the sentence \( x \) can be represented as \( I \in \mathbb{R}^{|x| \times d} \). Then the output of the \( k \)-th sliding window (with length \( l \)) of the convolutional layer is computed as:

\[ f_k = \tanh(W_c \cdot W_{m-l+1:m} + b_c), \]

where \( W_c \) and \( b_c \) are parameters, and \( W_{m-l+1:m} \) denotes the concatenation of \( l \) word embeddings within the \( m \)-th window in sentence \( x \). Supposing the number of filters we used is \( q \), for sentence \( x \), we can get its representation \( S \):

\[ f = [f_1, f_2, \cdots, f_{|x|-l+1}], \]

\[ u_i = \max\{f\}, \]

\[ S = [u_i]_{i=1}^{q}, \]

(12) (13) (14)

where \( f \in \mathbb{R}^{|x|-l+1} \) is the feature map, \( \max \) is the max-pooling operation to obtain the most salient feature \( u_i \). After that, we use the softmax classifier to get probability distribution over \( k + 1 \) classes as follows:

\[ D(x; \theta_d) = \text{softmax}(W_p \cdot S + b_p) = [D_1(x; \theta_d), \ldots, D_{k+1}(x; \theta_d)]. \]

(15)

### 3.2.5. Training

Due to the complexity of language and the infinite space of the arrangement of sequences, it is difficult to rely solely on the discriminator to guide the generation of generator from scratch. Therefore, we combine MLE pre-training and GAN goals, and the former can be regarded as a word-level fitness goal, while the latter is to ensure sentence-level goals, such as realism and overall sentiment.

We perform the adversarial training of generators and discriminator, and train them alternately, as shown in Algorithm 1.

**Algorithm 1** The adversarial training process in SentiGAN and C-SentiGAN.

**Input:** Generators \( \{G_i(k; \theta_{g,i-1}; \theta'_g)\}_{i=1}^{k} \); Discriminator \( D(x; \theta_d) \); Encoder \( E \) (for C-SentiGAN); Real text dataset with \( k \) types of sentiment, \( T = \{T_1, \ldots, T_k\} \);

**Conditional text, \( T_z \) (for C-SentiGAN);**

**Output:** Well-trained generators \( \{G_i(k; \theta_{g,i-1}; \theta'_g)\}_{i=1}^{k} \); 

1. Initialize \( \{G_i\}_{i=1}^{k} \), \( D \) with random weights;
2. Pre-train \( \{G_i\}_{i=1}^{k} \) using MLE on \( T \) by minimizing Eq. (2);
3. Generate fake texts \( F = \{F_i\}_{i=1}^{k} \) using \( \{G_i\}_{i=1}^{k} \);
4. Pre-train \( D(X; \theta_d) \) using \( \{T_1, \ldots, T_k, F\} \);
5. repeat
6. for \( g \)-steps do
7. Sample random noise \( z \) from \( P_z \) (for SentiGAN);
8. Get high-level representation \( z = E(T_z) \) from encoder \( E \) (for C-SentiGAN);
9. for \( i \) in \( 1 \sim k \) do
10. Generate fake texts using \( G_i(z; \theta'_g) \);
11. Calculate penalty \( V_{D_{g,i}} \) by Eq. (7);
12. Update \( G_i(z; \theta'_g) \) by minimizing Eq. (8);
13. end for
14. end for
15. for \( d \)-steps do
16. Generate fake texts \( F = \{F_i\}_{i=1}^{k} \) using \( \{G_i(k; \theta_{g,i-1}; \theta'_g)\}_{i=1}^{k} \);
17. Update \( D(X; \theta_d) \) using \( \{T_1, \ldots, T_k, F\} \) by minimizing Eq. (10);
18. end for
19. until Convergence
20. return ;

### 3.3. Analysis

3.3.1. The multi-class classification objective

In this section, we explain how our multi-class classification objective can force each generator to focus more on generating its own sentimental texts that are far from sentimental texts generated by other generators, thus it helps improve the sentiment accuracy of the generated texts.
Firstly, the optimal i-th generator can learn the distribution of the real texts with the i-th sentiment type. The objective function of the discriminator is shown in Eq. (10). Using \( \sum_{i=1}^{k+1} D_i = 1, D_i \in [0, 1], \forall i \), we obtain the optimal distribution learned by the discriminator \( D \):

\[
D_i(x; \theta_d) = \frac{P_{r_i}(x)}{\sum_{i=1}^{k} P_{r_i}(x) + P_g(x)}, \quad i = \{1, \ldots k\}
\]

(16)

\[
D_{k+1}(x; \theta_d) = \frac{P_g(x)}{\sum_{i=1}^{k} P_{r_i}(x) + P_g(x)}.
\]

(17)

In addition, generators minimize the penalty \( V_{D_i}^G(x) \) and can be rewritten as minimizing:

\[
E_{x \sim P_g} \log D_{k+1}(x) + \sum_{i=1}^{k} E_{x \sim P_{r_i}} \log (1 - D_{k+1}(x)).
\]

(18)

Then by using Eq. (10), generators’ goal is to minimize the following:

\[
E_{x \sim P_g} \left[ \frac{P_g(x)}{\sum_{i=1}^{k} P_{r_i}(x) + P_g(x)} \right] + \sum_{i=1}^{k} E_{x \sim P_{r_i}} \left[ \frac{P_{r_i}(x)}{\sum_{i=1}^{k} P_{r_i}(x) + P_g(x)} \right] - (k+1) \log(k+1)
\]

\[
= KL \left[ \sum_{i=1}^{k} P_{r_i}(x) || P_{\text{avg}}(x) \right] + \sum_{i=1}^{k} KL(P_{r_i}(x) || P_{\text{avg}}(x)) - (k+1) \log(k+1)
\]

(19)

where \( P_{\text{avg}}(x) = \frac{\sum_{i=1}^{k} P_{r_i}(x) + P_g(x)}{k+1} \), and \( KL \) means Kullback-Leibler divergence [64,65]. The above objective obtains its global minimum if \( P_{\text{g}} = P_{\text{r}_i} (i = 1, \ldots, k) \) with the objective value of \( -(k+1) \log(k+1) \). In the case of one generator \((k = 1)\), Eq. (19) reduces to the Jensen-Shannon divergence \((JS)\) with the minimum objective value of \( -\log4 \).

Secondly, while keeping \( \theta_d \) constant, the i-th generator aims to minimize the penalty \( (V_{D_i}^G) \) given by the discriminator. Under the constraint of \( \sum_{i=1}^{k+1} D_i = 1 \), it is equivalent to minimize \( \sum_{j=1, j \neq i}^{k+1} D_j(x; \theta_d) \). Intuitively, in order to get lower penalties from the discriminator, the texts generated by the i-th generator must be more consistent with the i-th sentiment type and be far away from other sentiment types.

3.3.2. The penalty-based objective

In this section, we introduce how the penalty-based objective forces generators to generate diversified examples rather than generate repetitive and ‘safe’ samples, and thus it helps improve the diversity and quality of generated texts. We compare the generator’s objective functions of GAN, SeqGAN and our SentiGAN as follows:

\[
J_G(X) = \begin{cases} 
E_{x \sim P_g} [-\log (D(x; \theta_d))] & \text{GAN} \\
E_{x \sim P_g} [\sum_{t=1}^{k+1} -\log (G(x_t; x_{0:t-1} ; \theta_g) D(x; \theta_d))] & \text{SeqGAN} \\
E_{x \sim P_g} [\sum_{t=1}^{k+1} G(x_t; x_{0:t-1} ; \theta_g) V(X)] & \text{SentiGAN} 
\end{cases}
\]

(20)

As can be seen, there are two main improvements in our objective function.

Firstly, our penalty-based objective can be considered as an approximate measure of wasserstein distance [66] which always provides meaningful gradients, even when the distributions of \( P_r \) and \( P_g \) do not overlap. The objective functions of the original GAN and SeqGAN generator can be regarded as \( E_{x \sim P_g} [-\log (D(x))] \), where \( P_g \) is the sample distribution generated by the generator. However, under the optimal discriminator \( D^* = \frac{P_g(x)}{P_r(x)+P_g(x)} \), the objective is:

\[
E_{x \sim P_g} [-\log (D^*(x))] = KL(P_g||P_r) - E_{x \sim P_g} [\log (1 - D^*(x))]
\]

\[
= KL(P_g||P_r) - 2JS(P_r||P_g) + 2\log2 + E_{x \sim P_g} [\log D^*(x)]
\]

\[
= KL(P_g||P_r) - 2JS(P_r||P_g)
\]

(21)

where \( JS \) is Jensen-Shannon divergence [67]. There are two serious problems with this goal of minimization of equivalence: 1) the first is that it minimizes the \( KL \) divergence of the generation distribution and the true distribution, but at the same time maximizes the \( JS \) divergence of them. It leads to the gradient instability problem. 2) Since the \( KL(P_g||P_r) \) is a symmetric metric, the generator prefers to generate repetitive but ‘safe’ rather than diversity samples because the second penalty is much larger than the first one. Thus it leads to the mode collapse problem.
Inspired by [66], we use the wasserstein distance of the two distributions to define the loss of the generator:

\[
W(P_r, P_g) = \frac{1}{K} \sup_{\|x\|_1 \leq K} |\mathbb{E}_{x \sim P_r}[\sigma(x)] - \mathbb{E}_{x \sim P_g}[\sigma(x)]|.
\] (22)

where the distribution function \( \sigma(x) \) is needed to satisfy Lipschitz continuity and its Lipschitz constant is \( K \). In this paper, we use the discriminator \( D \) to fit such a lipschitz function, and define it as \( \sigma(x) = D(x; \theta_d) \). As a consequence, Eq. (22) can be approximated as follows:

\[
\min_{G} \max_{D} W(P_r, P_g) \approx \frac{1}{K} (\mathbb{E}_{x \sim P_r}[D(x; \theta_d)] - \mathbb{E}_{x \sim P_g}[D(x; \theta_d)]).
\] (23)

Finally, our generator's objective is defined as Eq. (8) without the log function.

Secondly, we use penalty \( V(x) \) instead of reward \( D(x) \) like SeqGAN. Our penalty-based loss function \( G(x_t|x_{0:t-1}; \theta_g)\ V(x) \) can be thought of as adding \( G(x_t|x_{0:t-1}; \theta_g) \) to the reward-based loss function \( -G(x_t|x_{0:t-1}; \theta_g)D(x; \theta_d) \).

\[
G(x_t|x_{0:t-1}; \theta_g)\ V(x) = G(x_t|x_{0:t-1}; \theta_g)(1 - D(x; \theta_d))
\]

\[
= G(x_t|x_{0:t-1}; \theta_g) - G(x_t|x_{0:t-1}; \theta_g)D(x; \theta_d)
\] (24)

Therefore, our penalty-based loss function forces the generator to prefer a smaller \( G(x_t|x_{0:t-1}; \theta_g) \). Thus it results in the generation of diversified samples, rather than repetitive but 'good' samples.

4. Experiments

4.1. Datasets

For the generic text generation task, we use three kinds of real datasets for evaluation (statistics are shown in Table 1). Since generating very long texts is very challenging and we will study this problem in our future work, we simply refer to the work of [68] and focus on generating short sentences (length \( \leq 15 \) words) of two sentiment types (positive and negative).

**Movie Reviews (MR).** We use Stanford Sentiment Treebank [69] which has two sentiment classes (negative and positive). The original dataset has a total of 9,613 sentences. We select sentences containing at most 15 words, and the resulting dataset contains 2,133 positive sentences and 2,370 negative sentences.

**Beer Reviews (BR).** We use the data scraped from BeerAdvocate [70]. BeerAdvocate is a large online review community, and the data has 1,586,614 reviews for 66,051 distinct items composed by 33,387 users. Each review is accompanied by a number of numerical ratings, corresponding to 'appearance', 'aroma', 'palate', 'taste', and also the user's 'overall' impression. We select sentences containing at most 15 words, and the resulting dataset contains 1,437,767 positive sentences and 11,202 negative sentences.

**Customer Reviews (CR).** We use customer reviews of various products [71]. We select sentences containing at most 15 words, and the resulting dataset contains 1,024 positive sentences and 501 negative sentences.

For the conditional text generation task, unfortunately, large-scale emotional corpora with conditional input is difficult to come by. Without loss of generality, we choose one of the representative tasks in the field of conditional text generation, that is, emotional conversation generation. More specifically, given the context input and a desired sentiment type, our model needs to generate responses of the specified sentiment type.

**Emotional Conversations.** We use the emotional tweet conversation data set [17], which collects a large corpus of Twitter conversations that include emojis in the response and assume the emojis convey the underlying emotions of the sentence. We select emotions of three types: positive, negative, and neutral. After that, we randomly split the remaining corpus into 80,003/10,000/10,000 conversation pairs as train/validation/test sets, and the statistics are shown in Table 2.

4.2. Data processing

The data processing workflow of our model is shown in Fig. 6. The entire process can be divided into two phases: data preprocessing and model training. In the data preprocessing step, we first perform tokenization, stemming and lowercase conversion on text files. Then we get a vocabulary and then divide processed files by sentiment types. In the model training step, we first use MLE to pretrain all generators as shown in Sec 3.1.1, and then use Algorithm 1 to train generators and discriminator alternately.

---

**Table 1**

Statistics for MR, BR and CR datasets.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>#Train</th>
<th>#Test</th>
<th>#Vocab</th>
<th>#Max-Length</th>
<th>#Mean-Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>MR</td>
<td>2,133</td>
<td>2,370</td>
<td>5,000</td>
<td>6,399</td>
<td>15</td>
</tr>
<tr>
<td>BR</td>
<td>1,437,767</td>
<td>11,202</td>
<td>5,000</td>
<td>7,157</td>
<td>15</td>
</tr>
<tr>
<td>CR</td>
<td>1,024</td>
<td>501</td>
<td>5,000</td>
<td>2,306</td>
<td>15</td>
</tr>
</tbody>
</table>
4.3. Parameter setting

For generators, the input/hidden dimension size of single-layer LSTM-RNNs is set to 300, batch size is 32, and the maximum clipping gradient value is set to 5.0. The optimization algorithm is RMS-Prop [72] with learning rate 0.001.

For discriminator, the convolution layer is the same as [63] and the number of filters is set to 32 (q = 64). For the hyper parameters of Adam optimizer [73] in it, we set the learning rate $\alpha = 0.001$, two momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ respectively, and $\epsilon = 10^{-8}$.

For C-SentiGAN, the encoder $E$ is also a single-layer LSTM-RNNs with a hidden size of 300, and the global attention mechanism is Luong attention [53] with dot alignment function.

We train our model on each dataset respectively, and randomly initialize word embeddings with the dimension size of 300. The dropout rate [74] is set to 0.3 to prevent overfitting. The $N$ in Monte Carlo search is set as 15. In the pre-training step, we pre-train generators for 120 steps, and pre-train the discriminator for 50 steps. In adversarial training, the g-steps is 5 and d-steps is 1. We implement our model based on Tensorflow and use a TITAN X graphic card for learning.

4.4. Generic sentimental text generation

Without loss of generality, we set $k$ to 2 in the experiments and let SentiGAN generate two types of sentimental texts (positive and negative). We first use a state-of-the-art sentiment classifier [15] which achieves an accuracy of 90% on the SST test set, to automatically evaluate the sentiment accuracy of the generated texts. We compare with several state-of-art generic text generation methods, including RNNLM [22], GSGAN [25], SeqGAN [19], MaliGAN [24], RankGAN [23], Variational Autoencoders (VAE) [26], Conditional GAN (C-GAN) [30] and Semi-supervised VAE (S-VAE) [27]. It is worth noting that pre-training was used for all selected baselines.

4.4.1. Sentiment accuracy of generated texts

We use each model to generate 5K positive sentences and 5K negative sentences, which is trained on each of the above three datasets, respectively. The results are shown in Table 3. In order to investigate whether it is better to train with multiple generators than a single generator, we made a comparison with SentiGAN($k = 1$). Note that RNNLM, GSGAN, SeqGAN, MaliGAN, RankGAN, VAE, SentiGAN($k = 1$) cannot generate texts with two sentiment labels simultaneously, so we train each of these models on positive reviews and negative reviews, respectively.

---

**Table 2**

Statistics for emotional conversation dataset. ‘#Mean-L’ and ‘#Max-L’ refer to the average length and maximum length, respectively.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Positive</th>
<th>#Negative</th>
<th>#Neutral</th>
<th>#Overall</th>
<th>#Mean-L</th>
<th>#Max-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>28,467</td>
<td>27,155</td>
<td>24,381</td>
<td>80,003</td>
<td>16.632</td>
<td>30</td>
</tr>
<tr>
<td>Validation</td>
<td>3,228</td>
<td>3,481</td>
<td>3,291</td>
<td>10,000</td>
<td>17.823</td>
<td>25</td>
</tr>
<tr>
<td>Test</td>
<td>3,862</td>
<td>3,201</td>
<td>2,937</td>
<td>10,000</td>
<td>13.387</td>
<td>27</td>
</tr>
</tbody>
</table>

---

Note: The source codes of our methods are publicly available in [https://github.com/Nrgroup/SentiGAN](https://github.com/Nrgroup/SentiGAN).
4.4.2. MaliGAN, where fluency sentences.

From small can model with the dataset. Then we use manual evaluation to evaluate whether the sentiment types of the texts generated by various models are accurate. Specifically, we randomly extract 100 sentences from each of the generated positive and negative sentences and then ask three experts to rate each of them according to whether the sentence accords with the sentiment class of the dataset. The rating score ranges from 1 to 5, and 5 is the best. We finally take the average score across the sentences, as shown in Table 4.

From the comparison results in Table 3 and Table 4, we can see that our proposed model (SentiGAN(k = 2)) outperforms all other methods, including C-GAN and S-VAE. The accuracy achieved by our model is promisingly high, indicating that the framework with mixture of generators and one multi-class discriminator can make each generator to generate their own sentimental texts better. What's more, comparing SentiGAN(k = 2) with SentiGAN(k = 1) shows that multiple generators can help each other and thus greatly improve the sentiment accuracy of texts generated by each single generator. In addition, our model remains the leading result, even on the small CR dataset.

4.4.2. Quality of generated texts

Further, we use four other evaluation metrics to measure the quality of generated sentences from various aspects.

Fluency: Same as [19,23], we use the perplexity of language model calculations to measure the fluency of the generated sentences. In this study, we use a language modeling toolkit - SRILM [75] to calculate the perplexity. SRILM calculates the perplexity of generated sentences using the language model trained on respective corpus. The results are shown in Fig. 7. We can see that C-GAN and S-VAE are not good at keeping the fluency of sentences. However, our model maintains good fluency while generating texts of different sentiment labels, and it even significantly outperforms the existing models on the small CR dataset.

Novelty: Similar to [76], we want to investigate how different the generated sentences and the training corpus are. In other words, we want to see if the generator simply copies the sentence in the corpus instead of generating new ones. We calculate the novelty of each generated sentence $S_i$ as follows:

$$Novelty(S_i) = 1 - \max_{j=1}^{\lfloor R \rfloor} \varphi(S_i, R_j)$$  (25)

where $R$ is the sentence set of the training corpus, $\varphi$ is a kind of text similarity function, and here we use Jaccard similarity function [77]. The average values over generated sentences are shown in Table 5, we can see that RNNLM, GSGAN, SeqGAN, MaliGAN, RankGAN and VAE are not good at generating new sentences. On the contrary, our model performs exceptionally well, with the ability to generate sentences different from that in the training corpus.
Fig. 7. Comparison of fluency (perplexity) of generated sentences (Lower perplexity means better fluency).

Table 5
Comparison of novelty and diversity of generated sentences.

<table>
<thead>
<tr>
<th>Method</th>
<th>Novelty</th>
<th>Diversity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MR</td>
<td>BR</td>
</tr>
<tr>
<td>Real data</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>RNNLM</td>
<td>0.267</td>
<td>0.283</td>
</tr>
<tr>
<td>GSGAN</td>
<td>0.248</td>
<td>0.265</td>
</tr>
<tr>
<td>SeqGAN</td>
<td>0.298</td>
<td>0.328</td>
</tr>
<tr>
<td>MaliGAN</td>
<td>0.308</td>
<td>0.312</td>
</tr>
<tr>
<td>RankGAN</td>
<td>0.312</td>
<td>0.329</td>
</tr>
<tr>
<td>VAE</td>
<td>0.287</td>
<td>0.347</td>
</tr>
<tr>
<td>SentiGAN ((k = 1))</td>
<td>0.344</td>
<td>0.409</td>
</tr>
<tr>
<td>C-GAN</td>
<td>0.368</td>
<td>0.398</td>
</tr>
<tr>
<td>S-VAE</td>
<td>0.328</td>
<td>0.369</td>
</tr>
<tr>
<td>SentiGAN ((k = 2))</td>
<td><strong>0.395</strong></td>
<td><strong>0.427</strong></td>
</tr>
</tbody>
</table>

Fig. 8. Comparison of intelligibility of generated sentences by human evaluation.

**Diversity**: Inspired by [76], we want to see if the generator can produce a variety of sentences. Given a collection of generated sentences \( S \), we define the diversity of sentences \( S_i \) as follows:

\[
Diversity(S_i) = 1 - \max_{j \neq i} \{ \phi(S_i, S_j) \}
\]

where \( \phi \) is the similarity function (Jaccard similarity in this study). We calculate the maximum similarity between each sentence \( S_i \) and other sentences in the collection. The average values are shown in Table 5, and we can see that our model can generate a variety of sentences, while other models cannot ensure the diversity of generated sentences.

**Intelligibility**: Like [23,20], here we use human evaluation for evaluating the intelligibility of generated sentences. We randomly extract 100 sentences from the generated sentences and then ask three graduate students to rate each of them according to its intelligibility. The rating score ranges from 1 to 5, and 5 is the best. We finally take the average score across the sentences and the three students, as shown in Fig. 8. We can see that our model performs better than all other methods and our model can generate sentimental sentences with best intelligibility. Moreover, comparing the results on different datasets, we can see that more data can train better models with respect to intelligibility \( CR < MR < BR \).
Table 6
Example sentences generated by SentiGAN and C-GAN trained on MR.

<table>
<thead>
<tr>
<th></th>
<th>C-GAN</th>
</tr>
</thead>
</table>
| **Positive**  | give it credit, this is our’s brilliant. *(Unreadable)*  
good, bloody fun movie makes me smile every time to get on alien. *(Unreadable)*  
powerfully moving! *(Very short)* |
| **Negative**  | very bad comedy. *(Very short)*  
a mere shadow of its predecessors  
a timeless classic western dog ...  
one of those history movie traps *(Wrong sentiment)* |

SentiGAN(*k* = 2)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>
| **Positive**  | a fantastic finally, simply perfect masterpiece.  
one of the greatest movies i have ever seen.  
funny and entertaining, just an emotionally idea but it was pretty good.  
the best comedy is a science fiction, captain is like a comic legend. |
| **Negative**  | one of the most disturbing and sickening movies i have ever seen.  
a story which fails to rise above its disgusting source material.  
the comedy is nonexistent.  
this is a truly bad movie. |

4.4.3. Case study of SentiGAN

In Table 6, we show example sentences generated by SentiGAN(*k* = 2) and C-GAN trained on the MR dataset. From the examples, we can see some problems (e.g., unreadable, very short, wrong sentiment) with the sentences generated by C-GAN. Whereas, our proposed model produces sentences that are more readable, sentimentally accurate, with better quality, and longer than that of C-GAN.

4.5. Conditional sentimental text generation

Without loss of generality, we evaluate our C-SentiGAN on the emotional conversation generation task which is one of the representative tasks in the field of conditional text generation. That is, given the context input, we want to generate responses with three kinds of sentiments (i.e., positive, neutral and negative). For comparison, we implement the Seq2Seq model [28,29], which uses two RNNs to build the encoder-decoder framework with global attention mechanism [53] and is most commonly used in dialogue systems [9,78]. We also implement a Conditional Variational Autoencoder (C-VAE) [31] model to generate specific sentimental texts, which was proposed to incorporate conditioning option in the generation process.

4.5.1. Results of emotional conversation generation

For the generated responses, we evaluate them from three aspects, including sentiment accuracy, text quality, and context relevance. For sentiment accuracy, we evaluated it using a well-trained classifier [15] that achieves 87.4% and 88.2% (3-class) accuracy on the validation and test sets, respectively. For text quality, we use the fluency metric to measure the quality of the generated responses. As for context relevance, we also use a well-trained FastText Classifier [79] to determine if the response matches the context (using the original context-response pairs in the corpus as positive examples and scrambled context-response pairs as negative examples), which achieves accuracies of 70.4% and 73.5% on the validation and test sets, respectively.

We train on the training set, adjust the parameters on the validation set, and test the results on the test set. The results are shown in Table 7.

From the results, we can see that the sentiment accuracy of the traditional Seq2Seq model is not good. However, without loss of context relevance, our model not only maintains good text quality, but also improves the sentiment accuracy.
Table 8
Some example emotional responses generated by different models.

<table>
<thead>
<tr>
<th>Sentiment</th>
<th>Positive</th>
<th>Neutral</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>i need out of this house</td>
<td>i won’t tell anyone</td>
<td>let we know who</td>
</tr>
<tr>
<td>Seq2Seq</td>
<td>i love you girl.</td>
<td>a nfl investigation that’s fair</td>
<td>what a bad news.</td>
</tr>
<tr>
<td>C-GAN</td>
<td>thank u!</td>
<td>i watched it already</td>
<td>lemme let it go.</td>
</tr>
<tr>
<td>C-VAE</td>
<td>just saw it, good news</td>
<td>get used to it</td>
<td>bad news! don’t come back.</td>
</tr>
<tr>
<td>C-SentiGAN</td>
<td>we loved you! come again soon?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Content</td>
<td>i’s just worried that ikea is</td>
<td>of course you would support</td>
<td>well yeah that is the most likely</td>
</tr>
<tr>
<td>Seq2Seq</td>
<td>going to poach him now.</td>
<td>this</td>
<td></td>
</tr>
<tr>
<td>C-GAN</td>
<td>i’m just kidding</td>
<td>should been there, done!</td>
<td>sorry, daddy</td>
</tr>
<tr>
<td>C-VAE</td>
<td>it’s a good list</td>
<td>i want to know</td>
<td>do’t be too impatient</td>
</tr>
<tr>
<td>C-SentiGAN</td>
<td>it’s brilliant news!</td>
<td>i was thinking the same!</td>
<td>damn, it’s really depressing!</td>
</tr>
<tr>
<td>Content</td>
<td>this rain is ruining my birthday</td>
<td>happy birthday love ya</td>
<td>you can do it, bro</td>
</tr>
<tr>
<td>Seq2Seq</td>
<td>happy birthday love ya</td>
<td>happy birthday marion!</td>
<td></td>
</tr>
<tr>
<td>C-GAN</td>
<td>it looks very tasty</td>
<td>come most welcome</td>
<td>that’s not a first</td>
</tr>
<tr>
<td>C-VAE</td>
<td>i hope you enjoy your birthday</td>
<td>you’re welcome</td>
<td>what a bad story!</td>
</tr>
<tr>
<td>C-SentiGAN</td>
<td>happy birthday! hope it’s good</td>
<td>don’t worry no one does</td>
<td>so sad and horrible at the same</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>time!</td>
</tr>
</tbody>
</table>

Fig. 9. Loss curves during training procedure of SentiGAN. \(G_1\) is the positive text generator and \(G_2\) is the negative text generator. The upper column is the MLE loss \(L_{mle}\) in Eq. (2) during the pre-training step, and the lower column is the adversarial loss \(L_{adv}\) in Eq. (8) during the adversarial training process.

4.5.2. Case study of C-SentiGAN

We show some generated responses in Table 8. As can be seen from the table, the generated sentences of our model are more accurate in sentiment type, have higher quality and richer content.

4.6. Loss curves during training procedure

Since training multiple generators is very challenging, we want to know if the generator’s results are really getting better, not accidental. Without loss of generality, we show different loss curves during the SentiGAN’s training procedure in Fig. 9. In the pre-training step, the MLE loss \(L_{mle}\) drops very smoothly. However, in the adversarial training phase, the adversarial loss \(L_{adv}\) fluctuates greatly. But the overall trend is slowly declining, and thus we argue that the capability of generators is constantly getting stronger.

4.7. Analysis of model complexity

In this section, we analyze the complexity of our models. Table 9 shows the comparison of approximate parameter numbers of different models. The results show that the parameter numbers of our models and other models are on the same order of magnitude.
Table 9
Comparison of parameter numbers of different models. \#N, \#L, and \#P represent the numbers of submodels (i.e., generator or discriminator), layers and overall parameters, respectively.

<table>
<thead>
<tr>
<th>Task</th>
<th>Method</th>
<th>Generator</th>
<th>Discriminator</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#N</td>
<td>#L</td>
<td>#P</td>
</tr>
<tr>
<td>Generic generation</td>
<td>RNNLM</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>GSGAN</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>SeqGAN</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>MaliGAN</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>RankGAN</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>VAE(k = 1)</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>C-GAN(k = 1)</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>S-VAE(k = 2)</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Conditional generation (3-class)</td>
<td>Seq2Seq</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>C-GAN</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>C-VAE</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>C-SentiGAN</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 10
Comparison of time costs of different models. ‘h’ and ‘m’ refer to hour and minute, respectively. We test on an Nvidia GeForce TITAN X card.

<table>
<thead>
<tr>
<th>Methods</th>
<th>MR Training</th>
<th>MR Test</th>
<th>BR Training</th>
<th>BR Test</th>
<th>CR Training</th>
<th>CR Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-GAN</td>
<td>5 h</td>
<td>3 m</td>
<td>6 h</td>
<td>3 m</td>
<td>4 h</td>
<td>3 m</td>
</tr>
<tr>
<td>S-VAE</td>
<td>4 h</td>
<td>2 m</td>
<td>6 h</td>
<td>2 m</td>
<td>3 h</td>
<td>2 m</td>
</tr>
<tr>
<td>C-SentiGAN(k = 2)</td>
<td>5 h</td>
<td>3 m</td>
<td>8 h</td>
<td>4 m</td>
<td>3 h</td>
<td>4 m</td>
</tr>
</tbody>
</table>

Further, we show the time costs of SentiGAN(k = 2) and some baseline models in Table 10. We can see that our model does not differ a lot in terms of time costs and it takes 3 to 8 hours to train SentiGAN(k = 2) and 3 to 4 minutes to test on the three datasets.

4.8. Sensitivity study of metrics

In this section, we investigate the influence of the similarity function used in the novelty and diversity metrics. We compare several common text similarity functions, including Jaccard similarity (Jac) [77], Cosine similarity (Cos) [80] and Euclidean similarity (Euc) [81]. Given two word sets (i.e., BOW vectors) A and B, their calculation formulas are as follows:

\[
Jac(A, B) = \frac{|A \cap B|}{|A \cup B|} 
\]

(27)

\[
Cos(A, B) = \frac{A \cdot B}{\|A\| \|B\|} 
\]

(28)

\[
Euc(A, B) = \frac{1}{1 + \sqrt{\sum_{i} (A_i - B_i)^2}} 
\]

(29)

Without loss of generality, we watch the sensitivity study results on the generic sentimental text generation task. Fig. 10 shows the results of using different similarity functions in the novelty and diversity indicators. We can see that our model is superior to baseline models regardless of any similarity function. In this study, the text similarity function \(\phi\) empirically uses Jaccard similarity because we find that it has better discrimination and smaller computational complexity.

4.9. Influence of the number of sentiment types

In this section, we want to explore the influence of the number of sentiment types (k). Without loss of generality, we evaluate our SentiGAN on the Stanford Sentiment Treebank dataset. But here, we re-divide the movie reviews into 3 categories and 5 categories based on the associated sentiment values, respectively. The statistics for new datasets with 3-class and 5-class are shown in Table 11 and Table 12, respectively.

Comparison results between several models with various evaluation metrics on the 3-class sentiment dataset and the 5-class sentiment dataset are shown in Table 13 and Table 14, respectively. From the comparison results, we can see that:

1. As k increases, our model still maintains good sentiment accuracy, while other models decline faster.

2. The increase of
Fig. 10. Influence of the similarity function in novelty and diversity metrics on generic sentimental text generation task.

Table 11
Statistics for 3-class sentiment dataset.

<table>
<thead>
<tr>
<th>Sentiment value interval</th>
<th>[0, 0.333)</th>
<th>[0.333, 0.666)</th>
<th>[0.666, 1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Num</td>
<td>1350</td>
<td>1554</td>
<td>1599</td>
</tr>
</tbody>
</table>

Table 12
Statistics for 5-class sentiment dataset.

<table>
<thead>
<tr>
<th>Sentiment value interval</th>
<th>[0, 0.2)</th>
<th>[0.2, 0.4)</th>
<th>[0.4, 0.6)</th>
<th>[0.6, 0.8)</th>
<th>[0.8, 1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Num</td>
<td>587</td>
<td>1151</td>
<td>916</td>
<td>1120</td>
<td>729</td>
</tr>
</tbody>
</table>

Table 13
Comparison results on the 3-class sentiment dataset. ↓ means the smaller the better, and the other is the opposite.

<table>
<thead>
<tr>
<th>Evaluation</th>
<th>Sentiment accuracy</th>
<th>Sentiment relevance</th>
<th>Fluency↓</th>
<th>Novelty</th>
<th>Diversity</th>
<th>Intelligibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real data</td>
<td>0.702</td>
<td>–</td>
<td>78.2</td>
<td>–</td>
<td>0.753</td>
<td>–</td>
</tr>
<tr>
<td>C-GAN</td>
<td>0.602</td>
<td>3.871</td>
<td>105.1</td>
<td>0.362</td>
<td>0.716</td>
<td>3.908</td>
</tr>
<tr>
<td>S-VAE</td>
<td>0.612</td>
<td>3.901</td>
<td>95.1</td>
<td>0.351</td>
<td>0.712</td>
<td>4.013</td>
</tr>
<tr>
<td>SentiGAN</td>
<td><strong>0.661</strong></td>
<td><strong>4.281</strong></td>
<td><strong>91.0</strong></td>
<td><strong>0.408</strong></td>
<td><strong>0.720</strong></td>
<td><strong>4.402</strong></td>
</tr>
</tbody>
</table>

Table 14
Comparison results on the 5-class sentiment dataset. ↓ means the smaller the better, and the other is the opposite.

<table>
<thead>
<tr>
<th>Evaluation</th>
<th>Sentiment accuracy</th>
<th>Sentiment relevance</th>
<th>Fluency↓</th>
<th>Novelty</th>
<th>Diversity</th>
<th>Intelligibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real data</td>
<td>0.425</td>
<td>–</td>
<td>78.2</td>
<td>–</td>
<td>0.753</td>
<td>–</td>
</tr>
<tr>
<td>C-GAN</td>
<td>0.312</td>
<td>3.271</td>
<td>102.4</td>
<td>0.370</td>
<td>0.708</td>
<td>3.811</td>
</tr>
<tr>
<td>S-VAE</td>
<td>0.302</td>
<td>3.287</td>
<td>95.1</td>
<td>0.316</td>
<td>0.691</td>
<td>4.028</td>
</tr>
<tr>
<td>SentiGAN</td>
<td><strong>0.367</strong></td>
<td><strong>4.203</strong></td>
<td><strong>93.2</strong></td>
<td><strong>0.390</strong></td>
<td><strong>0.718</strong></td>
<td><strong>4.314</strong></td>
</tr>
</tbody>
</table>

k has little effect on the quality of generated sentences, probably because we have pre-trained all models. (3) Regardless of different settings, our model outperforms all other competitors with a large margin.
The performance comparison of different methods on the synthetic data in terms of the negative log-likelihood (NLL) scores.

<table>
<thead>
<tr>
<th>Method</th>
<th>MLE</th>
<th>SeqGAN</th>
<th>RankGAN</th>
<th>SentiGAN(k = 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLL</td>
<td>9.038</td>
<td>8.736</td>
<td>8.247</td>
<td>6.924</td>
</tr>
</tbody>
</table>

![Learning Curve](image)

Fig. 11. The illustration of learning curves. Dotted line is the end of pre-training.

<table>
<thead>
<tr>
<th>Evaluation</th>
<th>Topicality relevance</th>
<th>Fluency↓</th>
<th>Novelty</th>
<th>Diversity</th>
<th>Intelligibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNNLM</td>
<td>2.318</td>
<td>110.221</td>
<td>0.231</td>
<td>0.601</td>
<td>2.637</td>
</tr>
<tr>
<td>SeqGAN</td>
<td>2.164</td>
<td>98.253</td>
<td>0.298</td>
<td>0.641</td>
<td>2.865</td>
</tr>
<tr>
<td>VAE</td>
<td>2.421</td>
<td>108.271</td>
<td>0.281</td>
<td>0.621</td>
<td>2.908</td>
</tr>
<tr>
<td>SentiGAN(k = 1)</td>
<td><strong>3.231</strong></td>
<td><strong>82.554</strong></td>
<td><strong>0.344</strong></td>
<td><strong>0.711</strong></td>
<td><strong>3.284</strong></td>
</tr>
</tbody>
</table>

4.10. Validation of penalty-based objective

In this section, we want to separately verify the validity of our proposed penalty-based objective. Specifically, we use a synthetic data set to test our SentiGAN in the mere use of the penalty-based objective (i.e., SentiGAN(k = 1)). The synthetic data\(^2\) consists of a set of sequential tokens which can be seen as the simulated data comparing to the real-world language data. We use the Oracle model\(^2\) to generate 10,000 sequences as the training set. We compared our SentiGAN with various published methods (SeqGAN [19], RankGAN [23]) on this dataset, as shown in Table 15. And the learning curves are shown in Fig. 11. The results show the effectiveness of using a penalty-based objective, and our SentiGAN is better than the other models in capturing the dependency of the sequential tokens.

4.11. Exploration of descriptive text generation

In this section, we want to verify if our model can be applied to the generation of other types of texts other than sentimental texts. Thus we validate our model on the descriptive text corpus: HappyDB [82]. HappyDB is a corpus of 100,000+ crowd-sourced happy moments, which is thus considered a one-class dataset (K = 1). Its content is mainly about events that make people happy in the past, and 83% of corpora contain only one sentence. We only use those instances that contain one sentence, and we get a total of 83711 happy moments. It is worth noting that we have removed the limit on sentence length on this dataset. This can also be explored for an impact on sentence length. Without loss of generality, we only test our SentiGAN on the generic text generation task, and Table 16 shows the evaluation results of generated descriptive texts of different models.

From the results, we can see that our SentiGAN outperforms other models, especially in topicality. This shows that the performance of our model does not drop much when it migrates to descriptive texts. It indicates that our models are easy

\(^2\) The synthetic data and the oracle model (LSTM model) are publicly available at https://github.com/LantaoYu/SeqGAN.
to extend to other text generation tasks with specific attributes. Moreover, we do not limit the maximum length of the generated sentences here, which shows the robustness of our model with respect to text length.

5. Conclusion and future work

In this paper, we propose SentiGAN and C-SentiGAN, which can generate a variety of high-quality texts of different sentiment labels in a generic or conditional manner. Extensive experimental results and analysis demonstrate the efficacy of our models, which can outperform a variety of popular deep text generation models.

Our models first use GAN to directly guide the generation of sentimental texts, and they interact with multiple generators to further improve the sentiment accuracy and quality of the generated sentences. This is very helpful in guiding sentimental text generation in artificial intelligence systems and is easily migrated to the generation of other types of texts with specific attributes. Moreover, the qualitative or quantitative evaluation methods we proposed in this article provide some recommended evaluation options for text generation tasks.

In future work, we will try to further enhance the quality of generated texts, especially for longer text generation. A possible solution is to develop a hierarchical generation model or coarse-to-fine generation model which can generate higher-level sketch first and then generate each detailed sentence accordingly. We will also apply our model to generate texts with other kinds of labels (e.g., different writing styles), and study sentiment consistency in other specific tasks such as machine translation.

Declaration of Competing Interest

There is no competing interest.

Acknowledgements

This work was supported by National Natural Science Foundation of China (61772036) and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). We thank the anonymous reviewers for their helpful comments. Xiaojun Wan is the corresponding author.

References
