Neurocomputing xxx (xxxx) xxx

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Customizable text generation via conditional text generative adversarial network

Jinyin Chen*, Yangyang Wu, Chengyu Jia, Haibin Zheng, Guohan Huang

College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China

A R T I C L E I N F O

Article history:
Received 11 September 2018
Revised 17 November 2018
Accepted 28 December 2018
Available online xxx

Keywords:
Text generation
Variable length
Emotion label
Conditional text generative adversarial network

A B S T R A C T

Automatically generating meaningful and coherent text has many applications, such as machine translation, dialogue systems, BOT application, etc. Text generation technology has attracted more attention over the past decades. A bunch of excellent methods are proposed; however, there are still challenges to generate text rivals the real one by human, such as most machines output fixed length text, or can only generate text quite the same with the input training text. In this paper, we put forward a novel text generation system, called customizable conditional text generative adversarial network, which is capable of generating diverse text content of variable length with customizable emotion label. It is more convenient for generating actual original text with specific sensitive orientation. We propose a conditional text generative adversarial network (CTGAN), in which emotion label is adopted as an input channel to specify the output text, and variable length text generation strategy is put forward. After generating initial texts by CTGAN, to make the generated text data match the real scene, we design an automated word-level replacement strategy, which extracts the keywords (e.g. nouns) from the training texts and replaces the specific keywords in the generated texts. Finally, we design a comprehensive evaluation metric based on various text evaluations, called mixed evaluation metric. Comprehensive experiments on real-world datasets testify that our proposed CTGAN behaves better than other text generation methods, i.e., generated text are more real compared with the real text than other generation methods, achieving state-of-the-art generation performance.

1. Introduction

With the development of natural language processing (NLP) [1,2], generating meaningful and coherent text or sequence is a critical important task. Automatic text generation has attracted more attention since it has wider application in electrical commercial [3], typical speech generation [4], and robot service Bot [5,6], etc [7–9]. The general idea in the text generation technology is to estimate the distribution of sentences in the corpus and then use it to generate real text content. It has a wide applicate prospect, i.e., human-computer interaction dialogue [10], questions and answers [11], and machine translation [12], etc [13,14]. As well as assisting in the manual writing of various articles, such as helping journalists to automatically write and publish news, in the field of e-commerce [15]. Generation text can realize the automatic comment writing, help scholars to complete the preparation of some papers, and improve the efficiency of scientific research [16].

Automatic text generation has received more attention over the past decades. Some earlier works try to use statistical methods to generate text. Markov et al.,markov1971extension proposed Markov Chain model which can quickly generate a large amount of text, but generated texts are not as smooth as human written. There are other statistical methods for text generation [18], although most of them are easy to implement, the semantic meaning of generated sentences are less well preserved.

Quite recently, with the prevalence of deep models, more and more researchers focus on deep text generation methods. Hopfield [19] proposed the recurrent neural network (RNN) model, which is a natural generalization of feedforward neural networks to sequences. Although its generated texts have more potential meaning of the corpus, they still suffer from spelling errors and fixed length only. Cho et al. [20] proposed the Seq2Seq model, which is a special encode-decode model based on multilayered Long Short-Term Memory (LSTM) model [21–23]. The text generated by Seq2Seq is of high quality, but it fails to generate long text. Yu et al. [24] proposed the sequence generative adversarial networks (SeqGAN), using reinforcement learning [25] and generative adversarial network (GAN) [26] to generate higher quality text. Its excellent performance makes it a state-of-art text

* Corresponding author.
E-mail addresses: chenjinyin@zjut.edu.cn (J. Chen), 2111603080@zjut.edu.cn (Y. Wu), 201603090214@zjut.edu.cn (C. Jia), 201303080231@zjut.edu.cn (H. Zheng), 201603090106@zjut.edu.cn (G. Huang).

https://doi.org/10.1016/j.neucom.2018.12.092
0925-2312/© 2019 Elsevier B.V. All rights reserved.

generation method. In most cases, the deep model outperforms the traditional statistical method, but it still faces challenges, for example, more real text comparable with human (e.g. cannot be detected by fake review filter), high quality long text, minimal error spell and appointed emotional text.

Inspired by SeqGAN, we are interested in generating variable length text which can appoint emotion label and has controllable topic, while promising the quality of text content when detected by filters. In real world, the length of text written by human is usually unfixed, has various emotion (e.g. positive, negative, neutral), and belongs to clear topic. However, since most text generation methods ignore these features when working, which makes the huge gap between the machine generation and human writing.

In this paper, we propose a novel text generation method, named customizable text generation via conditional text generative adversarial network (CTGAN). The main contributions of our work are as follows.

- A novel conditional text generative adversarial network (CTGAN) is put forward. It is capable of generating variable length text with appointed emotion label.
- An automated word-level replacement strategy is adopted to guarantee the quality and diversity of the text. It extracts the keywords (e.g. nouns) from the training data set and replaces the specific keywords in the generated texts.
- Comprehensive text evaluation experiments are carried out to validate that the text generated by CTGAN are more real than state-of-art text generation method.

The rest of paper is organized as follows. In Section 2, we introduce the related work of text generation methods and text evaluations. In Section 3, we explain CTGAN in detail. In Section 4, we empirically evaluate CTGAN with other text generation methods on various real-world data sets belong to different application area. In Section 5, we conclude the paper and highlight future research direction.

2. Related works

2.1. Text generation methods

During the past decades, a tremendous amount of literature on text generation have emerged. In general, text generation methods can be roughly categorized into traditional method and deep model based method.

In the traditional method, Markov and O'leary [27] proposed the Markov Chain model for text generation. Markov chain has a wide range of application, and text generation is one of the most interesting. In such task, each state is part of the text, usually a word. State and transformation are generated by some corpora, then traverse the entire chain and output words for each state to generate text. The text generated in this way usually has no practical meaning, since the chain does not contain enough information to preserve any potential meaning and grammatical structure of the corpus [28]. Many statistical methods for text generation were proposed in past decades. However, the texts generated by most of them have less potential meaning of the corpus.

Considering limitations in the traditional text generation algorithms, the deep text generation algorithm is gradually developed to generate more reasonable text [29]. RNN model, a basic deep text generation model, was proposed by Hopfield [19]. It have both internal feedback connections and feedforward connections between processing units. The RNN model iteratively inputs the text for training, obtains the minimum effective confusion value and corresponding weight. Each time the occurrence probability of the next character is evaluated during the generation process, the highest one is generated until the sentence is complete. The text generated by RNN model is more in line with human writing habits and preserve more potential meaning of the corpus. Hochreiter and Schmidhuber [23] proposed a time recurrent neural network model, called Long Short-Term Memory (LSTM), which is a variant RNNs model that can process time-dependent data of long-term dependence. By increasing the input gate, forget gate and output gate, the weight of the self-loop in LSTM has been changing. So, LSTM model can dynamically change the integral scale when the model parameters are fixed, thus avoids the problem of gradient disappearance or gradient expansion. Cho et al. [20] proposed the Seq2Seq model, which uses a multilayered LSTM model to map the input sequence to a vector of a fixed dimensionality and another deep LSTM to decode the target sequence from the vector [10]. When generating each word, it is evaluated that the current generation process is copying the original text content or generating other content. Such model can implement the task of digest generation, input a long sentence and generate a general statement [30]. The Seq2Seq model can generate text of different lengths each time, and the generated text is of high quality. However, the generated texts are very short, and the training and generation process is cumbersome.

In order to generate more real text, Yu et al. [24] proposed the SeqGAN model to effectively train GAN model for text generation via policy gradient [31]. SeqGAN consists of a generator and a discriminator that form a GAN model that uses the idea of reinforcement learning to solve text generation problems [26,32]. The discriminator is a convolutional neural network (CNN) [33] whose basic structure is a feedforward neural network, while the generator is a LSTM model [34]. The purpose of the generator is to produce a real text. The purpose of the discriminator is to distinguish fake text generated by the generator and the real one. Throughout the training process, the generator uses the words in the Monte Carlo search [34] and selects each word to be generated based on the strategy function until a complete sentence is generated. The discriminator dynamically updates through the positive examples provided by the real text and the fake examples generated by the generator, while further improving the generated model. Once you have a set of near real generation sequences, you can retrain the discriminator. Iterate through the above steps until the generator can generate better text, and the discriminator can hardly distinguish between false text and real text before stopping training. Although the SeqGAN model improves the quality of text generation, it still cannot generate variable length text, and can only generate text of the same emotion label each time. In addition to SeqGAN, there are many methods to generate text or sequence based on GAN. Zhang et al. [35] proposed the TextGAN. Instead of using the standard objective of GAN, it matches the high-dimensional latent feature distributions of real and synthetic sentences, via a kernelized discrepancy metric. Guo et al. [36] proposed the LeakGAN model based on SeqGAN which address the problem for long text generation. It allows the discriminative model to leak its own high-level extracted features to the generative model to further improve the generative model.

2.2. Text evaluations

In order to evaluate the text generated by machine, various evaluation methods were proposed. In this section, we will introduce a bunch of metrics to estimate its performance.

- *Winnowing* [37]: Winnowing method is a widely used method to identify duplicate or near-duplicate texts. It uses window movement to select hash values and improve algorithm efficiency. After the window selects all the hash values, the same merge processing is performed, and finally a smaller number of hash value sets, that is, the obtained

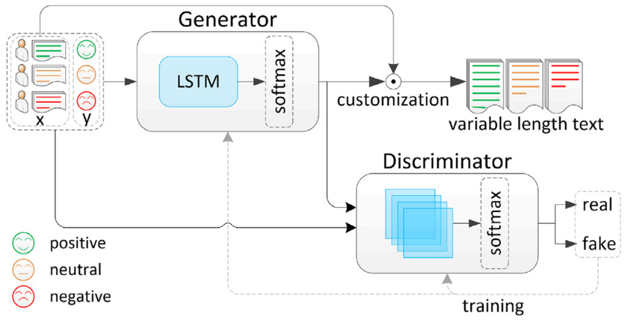


Fig. 1. The framework of CTGAN model,which consists of generator and discriminator. In the generator, LSTM module is applied to text generation and softmax is activation function. The input of generator is one-hot encoded text with emotional labels as conditions, and the output is generated text. In the discrimitor, CNN module is used for text feature extraction and softmax is the activation function. The input of discriminator are one-hot encoded real text and generated text with corresponding emotions as conditions, and the output are 'real' label and 'fake' label, respectively. After training, customization option is applied to generate more diverse variable length text.

document fingerprints are obtained. The similarity of two text fingerprints is calculated by the Jaccard similarity [38]. The Jaccard coefficient of the two texts is defined as the ratio of the size of the intersection of the two texts to the size of the union. The larger the value of the Jaccard coefficient, the higher the degree of similarity. In terms of grammar, the larger the value of the Jaccard coefficient between generated text and real text, the better performance of the text generation method. In terms of the meaning of the text, the smaller the value of the Jaccard coefficient between generated text and real text, the more diverse the generated text.

- *Cosine similarity* [39]: Cosine similarity is used to calculate the degree similarity between texts, and is also used to evaluate the diversity of text content. The texts that need to be similarly analyzed are segmented to obtain a list of words. The words in the list are one-hot encoded to get the number of occurrences of each participle. After the word frequency vector of two sentences is obtained, it becomes the cosine value for calculating the angle between the two vectors. The larger the value, the higher the similarity.
- *Parts-of-speech tagging (POS tagging)* [40,41]: POS tagging is used to count the word frequency of various parts of speech [40]. In this method, sequence annotation is performed using a hidden Markov model (HMM) [42], which is a statistical model used to describe a Markov process with implicit unknown parameters. The observable sequence in HMM is the participle of the given sentence in the part-of-speech tag, while the hidden state is the different part of speech. In order to achieve the part-of-speech tagging of sentences, it is necessary to first use the corpus to train an HMM, and then segment and mark the sentences.
- *SentiWordNet* [43]: The emotional tendency contained in the text is also a very important factor. We use retrieves emotional vocabulary from SentiWordNet for sentiment analysis. SentiWordNet is a vocabulary resource for opinion mining. SentiWordNet assigns three emotional scores to each of WordNet's synonyms: positivity, negativity, and objectivity.

3. Method

In this section, we introduce the framework of CTGAN model, as shown in Fig. 1. For convenience, the definitions of symbols used in this paper are briefly summarized in Table 1.

Table 1
The definitions of symbols.

Symbol	Definition
G_θ	θ-parameterized conditional generator
D_a	ϕ-parameterized conditional discriminator
P_{data}	Sample from real data
δ	The vocabulary of candidate tokens
$Y_{1:T} = (y_1, \ldots, y_T)$	Generated sequence with length T
$J(\theta)$	Objective function of generator
c	Condition
R_T	Reward from D_ϕ
$Q(s_0, y_1; c)$	Action-value function with state s_0 and taking action y_1
G_β	A roll-out policy in Monte Carlo search
$MC^{G_\beta}(N; Y_{1:t})$	An N-times Monte Carlo search
$Loss(D_\phi)$	The loss function of discriminator
h_c^t	Hidden state with condition c for x_t
g	Update function
s	Softmax activate function
$p(y_t \vert x_1, \ldots, x_t)$	Output token distribution
b	Bias vector
V	Weight matrix
$M_c^{1:T}$	A matrix that represents a input sequence $x_1, \ldots, x_t$
K	The size of candidate token δ
$\odot$	Concatenation operator
ω	Convolutional kernel
u	Windows size
e_c^i	A new feature map
ρ	Non-linear activate function
d	Bias
$\psi(g_t)$	The mixed score of the text g_t
ξ	The scale parameter of statistical evaluation metrics
μ	The scale parameter of hiddenness evaluation metrics
S	Statistical evaluation metrics
H	Hiddenness evaluation metrics
$\bar{\psi}(g_t)$	Mapped score of the text g_t
$\hat{\psi}(g_t)$	Average mixed score of real texts

- *Generate initial text*. CTGAN is consisted of a conditional LSTM as the text generator, and a fake text filter as the discriminator. In conditional LSTM, emotion label, representing the conditional channel, is input for target label of text along with real text as training data. Thus, appointed emotion texts are generated to guarantee both diversity and application demand. Discriminator is a Convolutional Neural Network (CNN) applied to classify real text and generated one. Based on the cross entropy of D and 1, we use the back propagation to train parameters of G with fixed D.
- *Review customization.* Generally, since the text is randomly sampled based on the word distribution, it is difficult to control specific information generation from the G model. For example, the name of a person in the movie, the name of dish in the restaurant, the address, etc. In order for better generation of text corresponding to target item, we customize the generated text by using specific keywords to further materialize the generated text. This information consists of specific nouns contained in the training text. Therefore, we propose an automated word-level replacement strategy, which extracts the target word from the training data set and replaces with the specific noun or adjective.

3.1. CTGAN model

Conditional text generative adversarial network (CTGAN) model is consisted of two designed modules, θ-parameterized conditional generator G_θ and ϕ-parameterized conditional discriminator D_ϕ. Before training CTGAN, we construct the vocabulary of candidate tokens δ based on all words in the training texts and add a special symbol "0" in δ. Then, we transfer the training texts into one-hot sequences based on the vocabulary of candidate tokens δ. We will show the details of CTGAN model as follow.

3.1.1. CTGAN via policy gradient

Since there is no intermediate reward, the objective of generator G_θ, generates a sequence $Y_{1:T} = (y_1, \dots, y_T)$ with condition c from the start state s_0 to maximize its expected end reward, can be described as follow:

$$J(\theta) = \mathbb{E}[R_T | s_0, \theta] = \sum_{y_1 \in \delta} G_\theta(y_1 | s_0; c) \cdot Q(s_0, y_1; c), \tag{1}$$

where R_T is the reward from the D_ϕ for a complete sequence. $Q(s_0, y_1; c)$ is the action-value function of a sequence under the condition c which the expected accumulative reward starting from state s_0, taking action y_1, and following policy G_θ. In general, the purpose of G_θ is to generate more real text which cannot be discriminated by D_ϕ from the true text written by human.

For G_θ, we need to consider how to estimate the action-value function. We use the REINFORCE algorithm [24] and D_ϕ to estimate probability of being real as the reward. The action function in condition c can be expressed as:

$$Q(y_T, Y_{1:T-1}; c) = D_\phi(Y_{1:T}; c), \tag{2}$$

For discriminator, it only provides a reward value for a fix-length sequence. In order to get long-term reward and evaluate the action-value for each state, we use Monte Carlo search [34] with a roll-out policy G_β which we set the same as the generator to sample the unknown last $T-t$ token. An N-times Monte Carlo search can express as:

$$MC^{G_\beta}(N; Y_{1:t}) = \{Y_{1:T}^1, \dots, Y_{1:T}^N\}, \tag{3}$$

where $Y_{1:T}^t$ $(t \in [1, \dots, N])$ is sampled according to the current state and G_β.

To get more accurate assessment of the action value, we use G_β starting from current state till the end of the sequence for N-times to get a fixed-length sequence. Then, action value can be redefined as follow:

$$Q(Y_{1:t-1}, y_t; c) = \begin{cases} \frac{1}{N}\sum_{n=1}^{N} D_\phi\left(Y_{1:T}^n; c\right), & t < T \\ D_\phi\left(Y_{1:T}^n; c\right) & t = T \end{cases} \tag{4}$$

where $Y_{1:T}^n \in MC^{G_\beta}(N; Y_{1:t})$.

In CTGAN model, the discriminator D_ϕ is used as the reward function, since it can be iteratively updated to improve the generator. When generating a more real sequence, discriminator will re-train based on the loss function of discriminator $Loss(D_\phi)$.

$$\begin{aligned} Loss(D_\phi) = min_\phi &- \mathbb{E}_{Y\sim p_{data}}[logD_\phi(Y; c)] \\ &- \mathbb{E}_{Y\sim G_\theta}[log(1 - D_\phi(Y; c))]. \end{aligned} \tag{5}$$

After training a new discriminator D_ϕ, we will update the generator in the next step. In the generator G_θ, its parameters θ are trained using gradient ascent, with the update role

$$\theta = \theta + \alpha \frac{\partial J(\theta)}{\partial \theta}, \tag{6}$$

where α is the corresponding learning rate, the gradient of the objective function is calculated as follow

$$\begin{aligned} \frac{\partial J(\theta)}{\partial \theta} = \mathbb{E}_{Y_{1:t-1}\sim G_\theta}\Bigg[&\sum_{y_t \in \delta} \frac{\partial G_\theta(y_t | Y_{1:t-1}; c)}{\partial \theta} \\ &\cdot Q(Y_{1:t-1}, y_t; c) \Bigg] \end{aligned} \tag{7}$$

3.1.2. Conditional generator

In CTGAN, the conditional generator is a LSTM with conditions which is designed to generate real sequences that are as similar as possible to the training sequences with the same condition. And the generator is shown in Fig. 2.

In LSTM model, it uses the update function g recursively to map the input sequence $\{x_1, \dots, x_t, \dots, x_T\}$ with condition c into a sequence of hidden states $\{h_c^1, \dots, h_c^t, \dots, h_c^T\}$ by using the update function g recursively.

$$h_c^t = g(h_c^{t-1}, x_t) \tag{8}$$

where hidden states are mapped into the output token distribution $p(y_t | x_1, \dots, x_t)$ by softmax activation function s.

$$p(y_t | x_1, \dots, x_t) = s(b + V h_c^t) \tag{9}$$

where b is a bias vector and V is a weight matrix.

In most of text generation methods, the texts are of fixed-length, which makes the difference between the generated one and the humans-written one. To overcome the gap, we design a special strategy for generating variable length texts, which involves three main steps as shown as follow:

- *Padding variable length training text with "0":* Since the texts in most of real text datasets are of variable length, we first pad all of training texts with "0" to make them into fixed-length.
- *Training the CTGAN model, and padding generated sequences with "0" in Monte Carlo search:* Generator is passed back to the intermediate action value via Monte Carlo search. Therefore, in the Monte Carlo search, we define that once the punctuation mark at the end of the sentence appears, the rest of words in the sequences are complemented by "0".
- *Removing "0" in the initial generated sequences:* In each of the initial generated sequences, the sequence is finished when "0" appears in it. Therefore, for initial generated sequences, we will remove words after "0", and output generated variable length sequences.

Although the variable length text is little shorter than the fixed length one, it can improve the quality while making it more in line with human writing characteristics.

3.1.3. Conditional discriminator

The conditional discriminator in CTGAN is a CNN model with conditions, which is designed to distinguish conditional generated sequence from the real sequence with the same condition and provide feedback. The framework of discriminator is shown as in Fig. 3.

Since the discriminator only provides a reward value for a fixed-length sequence, we consider pad the training sequences with "0" at the end of the sequence to make them equal in length. Then, all training sequences are one-hot type, a single input sequence $x_1, \dots, x_t$ with condition c can be represented as

$$M_c^{1:T} = x_1 \odot \dots \odot x_t \odot 0 \odot \dots 0 \tag{10}$$

where $\odot$ is the concatenation operator to build the matrix $M_c^{1:T} \in \mathbb{R}^{T\times K}$, where K is the size of the candidate tokens δ. The x_t is the K-dimensional token embedding. Parameter $\omega \in \mathbb{R}^{u\times K}$ applies a convolutional operation to a window size of u words to produce a new feature map:

$$e_c^i = \rho\left(w \otimes M_c^{i:i+u-1} + d\right) \tag{11}$$

where $\otimes$ is the summation of elementwise production, d is bias and ρ is a non-linear activate function. To extract more features, we utilize multiple number of kernels with different window size to extract features on $M_c^{1:T}$. Then, we use the max-over-time pooling operation to compress the feature map and extract main features on the feature maps. Finally, a fully connected layer with sigmoid activation is used to output the probability that the input sequence is real. As shown in Eq. (5), the optimization target of discriminator is to minimize the cross entropy between the real label and the predicted probability.

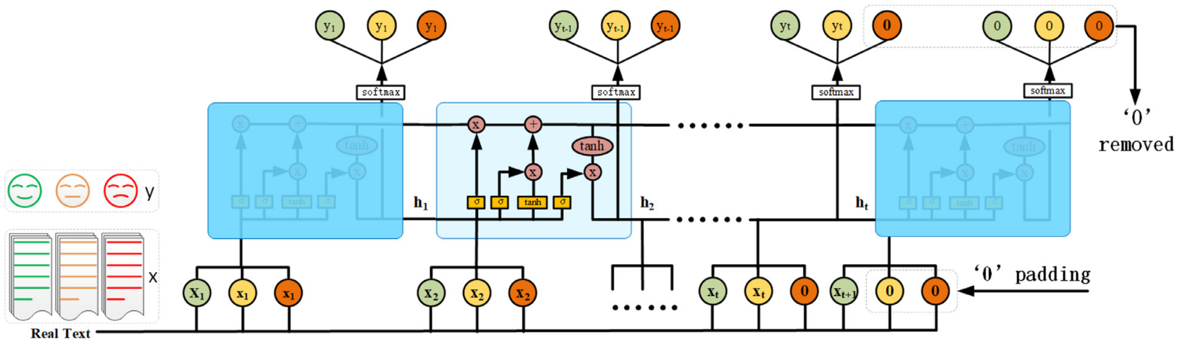


Fig. 2. The framework of generator. Training data sets are selected emotion labeled sequence of unfixed length, we use '0' to pad all training sequences so that all training sequences are fixed length. Finally, we modify the generated sequences according to the "0" symbol, so that the generated sequences can become more real variable length sequences.

Algorithm 1: The CTGAN model.

```
Input: Training texts with label $c \in [1, \dots, C]$, length of texts
       $T$, the epochs number of generator $g$-$step$, the epochs
       number of discriminator $d - step$.
Output: Generated texts.
1  Pad the training texts with "0", and transfer the training texts
   into one-hot sequences;
2  Initialize conditional generator $G_\theta$ and conditional
   discriminator $D_\phi$ with random weight;
3  Pre-train $G_\theta$, and set $\beta = \theta$;
4  Pre-train $D_\phi$ according to the generate initial fake sequences
   using $G_\theta$ and training sequences;
5  while CTAGN unconverges do
6      for g-step do
7          for $c \in 1 : C$ do
8              Generate a sequence $Y_{1:T} = (y_1, \dots, y_T)$ with
               condition $c$ by $G_\theta$;
9              for $t \in 1 : T$ do
10                 Compute $Q(Y_{1:t-1}, y_t; c)$ by Eq. (4)
11             end
12             Update generator parameters $\theta$ via policy gradient,
               and update $\beta$;
13         end
14     end
15     for $c \in 1 : C$ do
16         Generate fake sequences in condition $c$ by $G_\theta$ and
           combine with given training sequences with the same
           condition;
17         Train discriminator $D_\phi$ for $d - step$ epochs by Eq. (5).
18     end
19 end
20 return The texts which is generated by trained $G_\theta$.
```

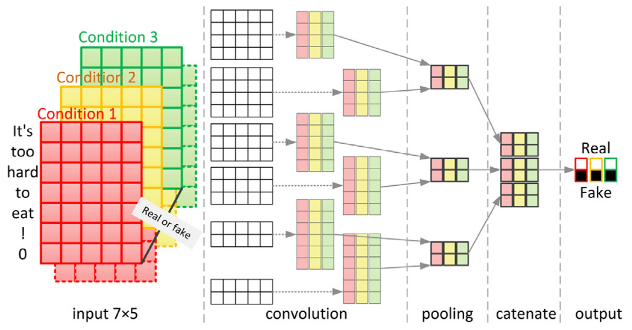
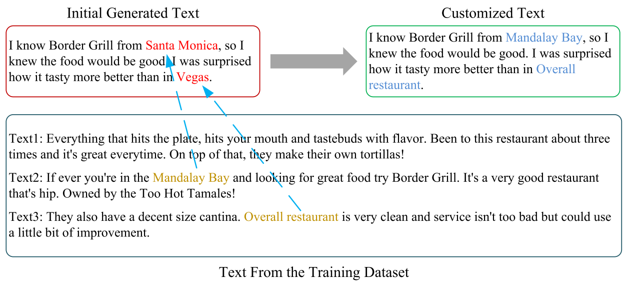


Fig. 3. The framework of discriminator. Take 'It's too hard to eat!' as an example, which is coded as one-hot and $k = 7$. The region size of the convolution kernel includes $4 \times t$, $3 \times t$ and $2 \times t$. After convolution, two feature maps for each region size are 4×1, 5×1 and 6×1, respectively. Feature maps with different colors correspond to different emotions. Then, max pooling is applied to unify the size of feature map to 1×1. And univariate vectora is concatenated and then connected to output layer.


Initial Generated Text

I know Border Grill from Santa Monica, so I knew the food would be good. I was surprised how it tasty more better than in Vegas.

Customized Text

I know Border Grill from Mandalay Bay, so I knew the food would be good. I was surprised how it tasty more better than in Overall restaurant.

Text1: Everything that hits the plate, hits your mouth and tastebuds with flavor. Been to this restaurant about three times and it's great everytime. On top of that, they make their own tortillas!

Text2: If ever you're in the Mandalay Bay and looking for great food try Border Grill. It's a very good restaurant that's hip. Owned by the Too Hot Tamales!

Text3: They also have a decent size cantina. Overall restaurant is very clean and service isn't too bad but could use a little bit of improvement.

Text From the Training Dataset

Fig. 4. The framework of text customization.

3.2. Text customization

In general, since the text is randomly sampled based on the word distribution, it is difficult to control the specific information generated from the CTGAN model. In this section, we further embody the generated comments by customizing the generated comments using specific keywords, which consists of specific words (e.g. nouns) included in the written comments, and propose an automated word-level replacement strategy.

This method works by replacing specific keywords in the initial generated texts with new words that better capture the context of the target item. This involves three main steps, as shown in Fig. 4.

- *Step 1, select the type of contextual information to be captured.* We choose a keyword to help identify context. For example, if you need to generate restaurant comments, the keyword can be "food," which captures food-related information. If the goal is to generate sports news, then the keywords can be "sports items" and the like.
- *Step 2, identify the words in the reviews of the training dataset that capture context.* We identify all words in the training dataset that are similar with the selected specific keywords. Word correlation is estimated by calculating lexical similarity using WordNet. We use a given similarity threshold to identify a set of words in the training dataset that have high lexical similarity to the specific keywords.

Food
Treated
Tell
Call
Family
Place
Great
Here
Good
Movie
Make
Film
Over
People
Time
Want
Child
America

(a) Yelp (b) Amazon (c) Film (d) Obama Speech

Fig. 5. Word frequency map of different data sets.

Table 2
Basic statistics of the datasets.

Datasets	#Texts number	#Classes	#Maximum SL	#Minimum SL
Yelp	924,675	5	44	1
Amazon	1126,696	3	48	1
Film	2000	2	60	2
Obama Speech	1000	1	74	2

- *Step 3, complete the replacement work*. Before replacement, we also find a set of words in the initial generated texts that have high lexical similarity to the specific keywords using the same method in the Step 2. Then, we replace the selected words in initial generated texts by sampling specific keywords selected in the Step 2.

4. Experiments

In order to testify the effectiveness of our CTGAN method, we compare it with some baseline text generation methods by performing a number of experiments on multiple real word datasets. Our experimental environment consists of i7-6700HQ 2.60GHzx8 (CPU), GTX 960M 8GiB (GPU), 8GB memory (DDR4) and Ubuntu 16.04 (OS).

4.1. Datasets

In this paper, we perform CTGAN model and multiple texts generation methods on various datasets, which is shown as follow. Then, we also show the word frequency map of all data sets in Fig. 5, and show the basic statistics of all datasets in Table 2, such as, texts number in dataset, number of classes, maximum and minimum of sentence length *SL*.

- *Yelp restaurant reviews* [44]. The dataset[1] is an open source dataset shared by the yelp website. The dataset contains reviews with 1–5 star, each of which has 82,030, 91,484, 142,318, 296,070, 312,773 reviews, respectively. In total, there are 924,675 reviews in Yelp dataset.
- *Amazon reviews* [45]. The data set is from the Amazon website[2] open source reviews. There are three types of Amazon reviews with different labels. They are clothing decoration, home kitchen and outdoor fitness, and each type of data set accounts for 278,677, 551,682, 296,337 reviews, respectively. In total, there are 1126,696 reviews in Amazon dataset.
- *Film review data* [46]. Film Review dataset[3] includes 1000 positive reviews and 1000 negative reviews. Each of the different film reviews has an emotional label (P/N).
- *Obama speech dataset* [24]. Obama Speech Dataset[4] has 1000 texts, and does not contain label information. It covers various aspects such as firearms, Middle East issues, economic and religious issues, etc.

4.2. Baseline methods

In order to evaluate the performance of CTGAN model, we compare our method with some text generation methods, such as, Markov Chain, RNN and Seq2Seq. We describe these text generation methods in detail as follow.

- *Markov Chain* [17]: Markov Chain, a basic traditional statistical model, uses an existing text dataset to construct a statistical model. Then, text can ge generated according to the constructed statistical model.
- *RNN* [19]: RNN is a neural network based on text data, that is, the current output of RNN is also related to its previous output.
- *Seq2Seq* [20]: In Seq2Seq, one LSTM model is used to map the input text to a vector of a fixed dimensionality, and then another LSTM model is used to decode the target text from the vector.

In this section, we select 200 texts in each of categories on all datasets as the training texts, and each of text generation methods will also generate the same amount of texts with conditions. In particularly, we set the length of texts T=30, the epochs number of generator g-step=1, and the epochs number of discriminator d-step=5.

4.3. Evaluation metrics

In this section, based on Section 2, we introduce some metrics to analyze generated texts from various aspects, such as, plagiarism detection, text diversity, word frequency and emotional tendency. We use Winnowing to calculate the similarity between generated texts and training texts, analyze text diversity by cosine similarity, analyze the word frequency by POS tagging, and analyze the subjective and sentimental of generated texts by SentiWordNet.

In particular, we also analyze average sentence length of the generated texts. The average sentence length is expressed as the average of the number of words in each sentence. Often, longer texts and comments give people a more real feel [47], so longer average length of generated texts mean better performance of text generation methods.

Based on above evaluation metrics, we divide them into two categories: statistic-based evaluation metrics and unstatistic-based

[1] https://www.kaggle.com/yelp-dataset/yelp-dataset
[2] http://jmcauley.ucsd.edu/data/amazon/
[3] http://www.cs.cornell.edu/people/pabo/movie-review-data/
[4] https://github.com/samim23/obama-rnn

Table 3
Average sentence length on all datasets.

Datasets	Average sentence length				
	Markov Chain	Seq2Seq	RNN	CTGAN	Ground truth
Yelp	12.14	8.67	13.78	13.26	13.74
Amazon	12.69	9.24	13.57	13.30	14.13
Film	12.56	9.91	13.52	14.42	14.21
Obama speech	13.64	10.88	14.16	14.95	15.14

Table 4
The frequency of three keywords in the generated texts and real texts.

Datasets	Markov Chain			Seq2Seq			RNN			CTGAN			Ground truth		
	Noun (%)	Verb (%)	Adjective (%)	Noun (%)	Verb (%)	Adjective (%)	Noun (%)	Verb (%)	Adjective (%)	Noun (%)	Verb (%)	Adjective (%)	Noun (%)	Verb (%)	Adjective (%)
Yelp	29.84	20.01	10.88	29.31	21.41	10.96	27.34	17.11	12.81	27.99	17.12	13.06	27.92	17.63	12.74
Amazon	28.82	19.47	10.13	29.06	20.11	10.64	28.01	16.04	13.17	28.91	16.23	12.98	28.43	16.80	12.56
Film	27.15	20.39	9.22	27.35	22.04	10.07	25.20	17.26	11.65	25.86	17.89	10.44	25.71	18.15	10.97
Obama Speech	28.55	18.90	9.97	28.74	19.75	10.29	26.17	17.31	11.85	27.03	17.20	11.63	26.68	17.99	11.36

evaluation metrics. Statistic-based evaluation metrics can be obtained through basic statistical information on texts, such as, average sentence length and word frequency. While unstatistic-based evaluation metrics can be obtained by analyzing further information in the generated texts, such as, subjective of generated texts, sentimental of generated texts and text diversity.

According to above two definitions, we propose a novel evaluation metrics, called mixed evaluation metric, which considers both statistical evaluation metrics S and hiddenness evaluation metrics H. The value of mixed evaluation metric $\psi(g_t)$ for generated text g_t as follow:

$$\psi(g_t) = \xi \sum S(g_t) + \mu \sum H(g_t), \tag{12}$$

where ξ represents the scale parameter of statistical evaluation metrics, μ represents the scale parameter of hiddenness evaluation metrics. Since both statistical evaluation metrics and hiddenness evaluation metrics are important in the evaluation of generated texts, we set both ξ and μ to 0.5. The larger $\psi(g_t)$ corresponds to the better generate effect.

4.4. *Result*

Now, let us present the results obtained by CTGAN and the baseline methods.

4.4.1. *Compare generated texts via various evaluation metrics*

In this part, we analyze the texts generated by different methods via various evaluation metrics which are described at Sec. 4.3. In particular, since texts in Amazon dataset and Obama Speech dataset are not labeled with emotion, we can not analyze the subjective and sentimental of texts in these datasets.

At first, we analyze texts generated by four text generation methods and the training texts on average sentence length, and show the statistics results in Table 3. On average sentence length, we can find that the texts generated by RNN and CTGAN are more similar with the ground truth texts than the texts generated by other text generation methods.

The second, we use POS tagging to analyze the frequency of special keywords (e.g. noun, verb and adjective) of all generated texts and real texts in Table 4. From Table 4, we can find that the word frequency of the texts generated by CTGAN are more similar with the ground truth texts in most cases, especially the frequency of noun.

Then, we use Winnowing to calculate the similarity between generated texts and real texts. In Fig. 6, the similarity of CTGAN and RNN model is around 0.1–0.2, while the similarity of Markov Chain and Seq2Seq model is around 0.03–0.06. So, in terms of grammar, we can find the texts generated by CTGAN and RNN are more real, and texts generated by Markov Chain and Seq2Seq are different with human written text and preserve less semantic meaning.

Next, we evaluate the diversity of generated texts of four text generation methods via cosine similarity on all datasets. In Fig. 7, we can see that the CTGAN outperforms all the other text generation methods in diversity, while Markov Chain method and Seq2Seq model perform worst in diversity. This is reasonable, since text customization in CTGAN model make the generated texts more diverse.

Finally, in Figs. 8 and 9, we analyze the sentimental and subjective of the generated texts, respectively. The results are shown in Figs. 8 and 9, for different datasets, where we can find that CTGAN model performs best, and the emotional trend of the texts generated by CTGAN are most similar to the emotional trend of real texts with the same condition.

4.4.2. *Filter generated text via mixed evaluation metric*

Based on the definition of mixed evaluation metric, we use this evaluation metric to evaluate the performance of various text generation methods. At first, we use real texts and the texts generated by one text generation methods for training to choose optimal values for ξ and μ. Then, we use the selected optimal parameters to evaluate the mixed score of the texts generated by all text generation methods based on Eq. (12). Next, we map all mixed score to the same range. The mapping function is $\bar{\psi}(g_t) = \frac{\psi(g_t)}{\bar{\psi}_r}$, where $\bar{\psi}(g_t)$ is mapped mixed score for texts g_t, ψ is the original mixed score, $\bar{\psi}_r$ is the average mixed score of real texts. Finally, we classify the generated texts and real texts according to a given mapped mixed score threshold, and use accuracy to evaluate the classification results. In this part, we only consider do experiments in the text data of the Yelp dataset.

In Table 5, we can find that the texts generated by CTGAN are more difficult to be identified in most cases. That is to say, Among all the generated texts, the text generated by CTGAN is most similar to the real text. The average accuracy of CTGAN is 0.56, the average accuracy of Markov Chain is 0.66, the average accuracy of Seq2Seq is 0.65, and the average accuracy of RNN is 0.57. The CTGAN reduces the accuracy by 1.78–17.86%.

The classification accuracy of various filters on generated texts and real texts with only statistical information or hiddenness in-

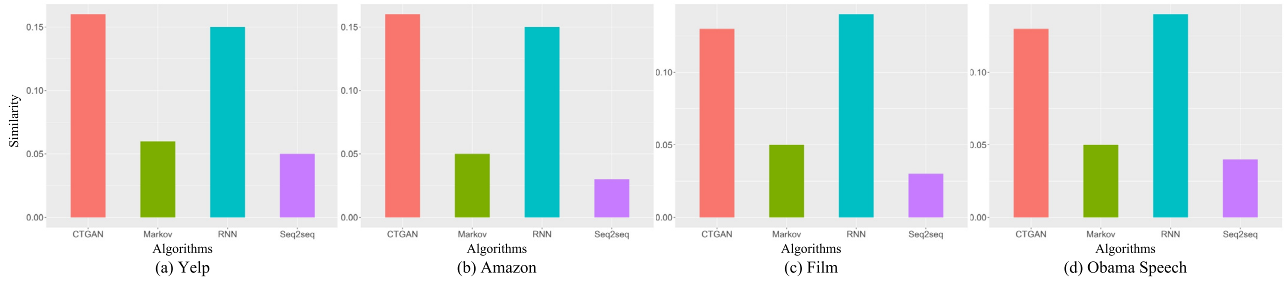


Fig. 6. Evaluate similarity of texts generated by variant text generation methods on all datasets.

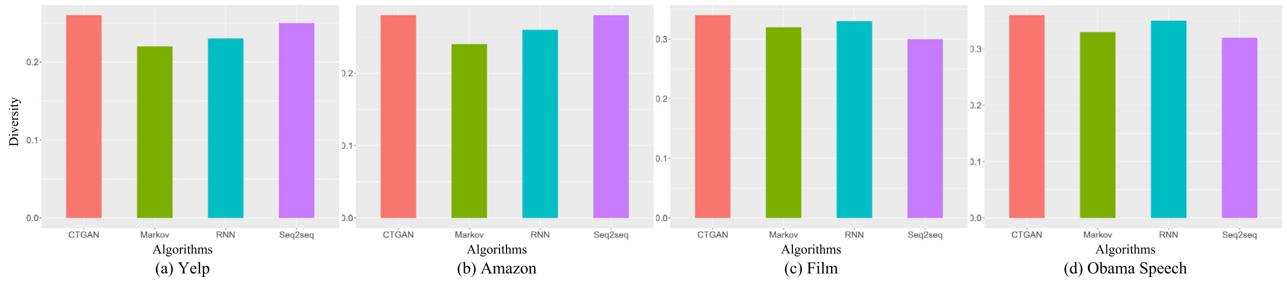


Fig. 7. Evaluate diversity of texts generated by variant text generation methods on all datasets.

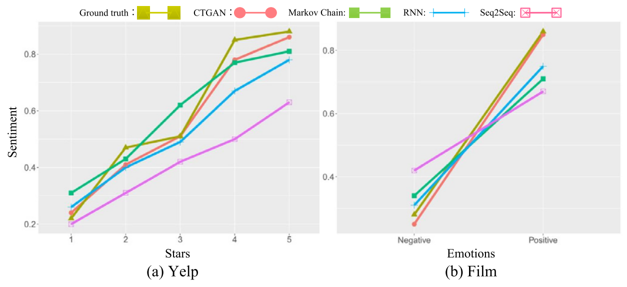


Fig. 8. Evaluate sentiment of texts generated by variant text generation methods on Yelp dataset and Film dataset.

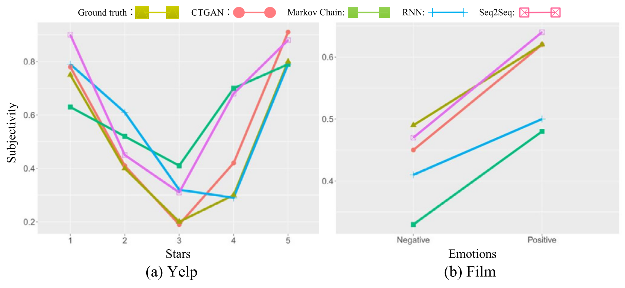


Fig. 9. Evaluate subjectivity of texts generated by variant text generation methods on Yelp dataset and Film dataset.

formation are shown in Tables 6 and 7, where we can see that these results are almost consistent with those in Table 5, i.e, the text generated by CTGAN is most similar to the real text. Surprisingly, from the results shows in these table, we can find that, in most cases, the filters with mixed evaluation metric perform better that the filters with only statistical information or hiddenness information.

Table 5
Accuracy of various filters on generated texts and real texts with different threshold.

Filters	Methods	Threshold				
		0.60	0.70	0.80	0.90	1.00
Markov Chain generated text for training	Markov Chain	0.57	0.58	0.63	0.72	0.80
	Seq2Seq	0.56	0.58	0.64	0.67	0.73
	RNN	0.52	0.54	0.54	0.54	0.70
	CTGAN	**0.51**	**0.53**	**0.53**	**0.54**	**0.69**
Seq2Seq generated text for training	Markov Chain	0.57	0.58	0.63	0.71	0.78
	Seq2Seq	0.57	0.58	0.63	0.69	0.76
	RNN	0.52	**0.54**	0.54	**0.53**	0.70
	CTGAN	**0.52**	**0.54**	**0.52**	**0.53**	**0.69**
RNN generated text for training	Markov Chain	0.56	0.60	0.63	0.72	0.80
	Seq2Seq	0.57	0.60	0.63	0.71	0.78
	RNN	0.52	0.54	0.53	**0.54**	0.71
	CTGAN	**0.51**	**0.51**	**0.51**	**0.54**	**0.70**
CTGAN generated text for training	Markov Chain	0.57	0.60	0.63	0.72	0.80
	Seq2Seq	0.57	0.59	0.63	0.71	0.80
	RNN	0.52	0.53	0.53	0.54	**0.71**
	CTGAN	**0.51**	**0.51**	**0.50**	**0.53**	**0.71**

Table 6
Accuracy of various filters on generated texts and real texts based on statistical information.

Filters	Methods	Threshold				
		0.60	0.70	0.80	0.90	1.00
Markov Chain generated text for training	Markov Chain	0. 51	0.54	0.60	0.70	0.77
	Seq2Seq	0.52	0.54	0.61	0.69	0.76
	RNN	0.50	0.52	0.53	0.54	0.68
	CTGAN	**0.44**	**0.49**	**0.52**	**0.53**	**0.65**
Seq2Seq generated text for training	Markov Chain	0.49	0.55	0.60	0.71	0.75
	Seq2Seq	0.49	0.52	0.56	0.67	0.72
	RNN	0.47	0.50	0.53	0.56	0.69
	CTGAN	**0.45**	**0.48**	**0.51**	**0.53**	**0.64**
RNN generated text for training	Markov Chain	0.53	0.57	0.62	0.70	0.77
	Seq2Seq	0.55	0.57	0.62	0.70	0.77
	RNN	0.50	0.53	0.54	0.60	0.71
	CTGAN	**0.48**	**0.49**	**0.50**	**0.52**	**0.70**
CTGAN generated text for training	Markov Chain	0.52	0.58	0.61	0.70	0.77
	Seq2Seq	0.54	0.56	0.61	0.70	0.76
	RNN	**0.48**	0.52	0.53	0.54	0.67
	CTGAN	**0.48**	**0.50**	**0.50**	**0.51**	**0.67**

Table 7
Accuracy of various filters on generated texts and real texts based on hiddenness information.

Filters	Methods	Threshold				
		0.60	0.70	0.80	0.90	1.00
Markov Chain generated text for training	Markov Chain	0.48	0.54	0.59	0.70	0.74
	Seq2Seq	0.48	0.52	0.58	0.66	0.71
	RNN	0.46	0.49	0.52	0.54	0.67
	CTGAN	**0.45**	**0.48**	0.52	**0.53**	**0.66**
Seq2Seq generated text for training	Markov Chain	0.48	0.53	0.59	0.71	0.76
	Seq2Seq	0.47	0.51	0.55	0.67	0.72
	RNN	0.46	0.49	0.52	0.53	0.69
	CTGAN	**0.44**	**0.47**	**0.50**	**0.51**	**0.64**
RNN generated text for training	Markov Chain	0.50	0.54	0.61	0.74	0.78
	Seq2Seq	0.50	0.55	0.60	0.69	0.77
	RNN	0.49	0.51	0.52	0.54	**0.70**
	CTGAN	**0.48**	**0.52**	**0.54**	**0.59**	**0.70**
CTGAN generated text for training	Markov Chain	0.51	0.54	0.61	0.73	0.77
	Seq2Seq	0.52	0.56	0.60	0.70	0.77
	RNN	0.48	0.54	0.55	0.63	0.71
	CTGAN	**0.47**	**0.48**	**0.49**	**0.53**	**0.70**

Table 8
The results of text classification on the texts generated by CTGAN.

Datasets	Classes	Accuracy	Recall	F1-score
Yelp	Star 1	1.00	1.00	1.00
	Star 2	1.00	0.43	0.60
	Star 3	0.00	0.00	0.00
	Star 4	0.40	1.00	0.57
	Star 5	1.00	0.67	0.80
Film	Negative	0.75	0.71	0.73
	Positive	0.72	0.76	0.74
Amazon	Clothing	1.00	1.00	1.00
	Home	0.88	1.00	0.93
	Sports	1.00	0.86	0.92

4.4.3. Conditional texts generated by CTGAN

In this part, we evaluate the conditional texts generated by the CTGAN model. Based on different conditions, the CTGAN model will generate different texts. So in Table 8, we evaluate the conditional texts generated by CTGAN model according to a simple text classification [48] which is trained based on the real texts with various conditions. The classification results will be shown based on accuracy, recall and F1-score in Table 8.

In Table 8, we can find that the trained text classification perform well on most of conditional texts generated by CTGAN model. Surprisingly, for the datasets of Amazon dataset, all classification results close to 100%, since the difference between the different label texts in Amazon dataset is larger. From the classification results on the Yelp dataset, we can find that the more classes in texts dataset, the harder it is to classify.

5. Conclusion

In this work, we put forward CTGAN for customizable text generation, for the possibility of generating a large amount of text using deep learning. In order to improve the quality of the generated text in the traditional RNN model, CTGAN can be trained on the comprehensive dataset with the standard label and generated according to the standard label setting. Different types and required texts simplify the text generation process to a certain extent, and it is more convenient to generate diverse text content. We specifically show how CTGAN generate text and how to effectively improve its authenticity, especially for restaurant reviews and news content. Finally, through several evaluation indicators of the generated text, the results were compared to highlight the superiority of CTGAN. The generated text is filtered separately using a self-designed fake text filter, and CTGAN performs better by filters compared with baselines. For future work, we plan to apply CTGAN in more natural language process applications, such as, dialogue systems and machine translation.

Conflict of interest

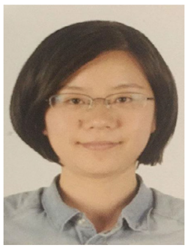
None.

Acknowledgments

This research was supported by Zhejiang Provincial Natural Science Foundation of China under Grant No. LY19F020025, Major Special Funding for “Science and Technology Innovation 2025” in Ningbo under Grant No. 2018B10063.

References

[1] G.G. Chowdhury, Natural language processing, Ann. Rev. Inf. Sci. Technol. 37 (1) (2003) 51–89.
[2] E. Liddy, Natural language processing, Encyclopedia of library and information science, 2nd ed, Decker, New York, NY, 2001.
[3] A. Ramos-Soto, A.J. Bugarin, S. Barro, J. Taboada, Linguistic descriptions for automatic generation of textual short-term weather forecasts on real prediction data, IEEE Trans. Fuzzy Syst. 23 (1) (2015) 44–57.
[4] J.R. Bellegarda, Combined statistical and rule-based part-of-speech tagging for text-to-speech synthesis: U.S. Patent 8,719,006[P]. 2014-5-6.
[5] S.A. Abdul-Kader, J. Woods, Survey on chatbot design techniques in speech conversation systems, Int. J. Adv. Comput. Sci. Appl. 6 (7) (2015).
[6] S. Reed, Z. Akata, H. Lee, B. Schiele, Learning deep representations of fine-grained visual descriptions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 49–58.
[7] H.-X. Hu, W. Yu, Q. Xuan, C.-G. Zhang, G. Xie, Group consensus for heterogeneous multi-agent systems with parametric uncertainties, Neurocomputing 142 (2014) 383–392.
[8] H.-X. Hu, G. Wen, W. Yu, Q. Xuan, G. Chen, Swarming behavior of multiple euler-lagrange systems with cooperation-competition interactions: an auxiliary system approach, IEEE Trans. Neural Netw. Learn. Syst. (99) (2018) 1–12.
[9] H.-x. Hu, W. Yu, G. Wen, Q. Xuan, J. Cao, Reverse group consensus of multi-agent systems in the cooperation-competition network, IEEE Trans. Circ. Syst. I Regul. Papers 63 (11) (2016) 2036–2047.
[10] V.K. Tran, L.M. Nguyen, Neural-based natural language generation in dialogue using rnn encoder-decoder with semantic aggregation. arXiv preprint arXiv:1706.06714, pp. 1–10, 2017.
[11] A. Kannan, K. Kurach, S. Ravi, T. Kaufmann, A. Tomkins, B. Miklos, G. Corrado, L. Lukacs, M. Ganea, P. Young, et al., Smart reply: Automated response suggestion for email, in: Proceedings of the Twenty-second ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 955–964.
[12] Z. Yang, W. Chen, F. Wang, et al. Improving neural machine translation with conditional sequence generative adversarial nets. arXiv preprint arXiv:1703.04887, pp. 1–10, 2017.
[13] X. Zhang, M. Lapata, Chinese poetry generation with recurrent neural networks, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 670–680.
[14] A. Bartoli, A. De Lorenzo, E. Medvet, D. Morello, F. Tarlao, “Best dinner ever!!!”: automatic generation of restaurant reviews with LSTM-RNN, in: Proceedings of the 2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI), IEEE, 2016, pp. 721–724.
[15] N.L. Latar, The robot journalist in the age of social physics: the end of human journalism? in: The New World of Transitioned Media, Springer, 2015, pp. 65–80.
[16] E. Wong, T. Liu, L. Tan, Clocom: mining existing source code for automatic comment generation, in: Proceedings of the 2015 IEEE 22nd International Conference on Software Analysis, Evolution and Reengineering (SANER), IEEE, 2015, pp. 380–389.
[17] A. Markov, Extension of the limit theorems of probability theory to a sum of variables connected in a chain, Dynamic probabilistic systems, vol. Markov models (1971) 552–577.
[18] G.A. Şeker, G. Eryiğit, Extending a crf-based named entity recognition model for turkish well formed text and user generated content 1, Semant. Web 8 (5) (2017) 625–642.
[19] J.J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, Proc. Natl. Acad. Sci. 79 (8) (1982) 2554–2558.
[20] K. Cho, B. Van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, Comput. Sci. (2014).
[21] K.S. Tai, R. Socher, C.D. Manning, Improved semantic representations from tree-structured long short-term memory networks. arXiv preprint arXiv:1503.00075, pp. 1–11, 2015.
[22] G. Kim, H. Yi, J. Lee, Y. Paek, S. Yoon, in: Lstm-based system-call language modeling and robust ensemble method for designing host-based intrusion detection systems, 2016 arXiv:1611.01726.
[23] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[24] L. Yu, W. Zhang, J. Wang, Y. Yu, Seqgan: Sequence generative adversarial nets with policy gradient, in: Proceedings of the AAAI, 2017, pp. 2852–2858.
[25] R.J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Mach. Learn. 8 (3–4) (1992) 229–256.
[26] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Proceedings of the International Conference on Neural Information Processing Systems, 2014, pp. 2672–2680.
[27] J.M. Conroy, D.P. O'leary, Text summarization via hidden markov models, in: Proceedings of the Twenty-forth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2001, pp. 406–407.
[28] R. Lebret, D. Grangier, M. Auli, Neural text generation from structured data with application to the biography domain, (2016) arXiv:1603.07771.
[29] B. Dong, X. Wang, Comparison deep learning method to traditional methods using for network intrusion detection, in: Proceedings of the IEEE ICCSN, 2016, pp. 581–585.
[30] W. Fedus, I. Goodfellow, A.M. Dai, Maskgan: Better text generation via filling in the. arXiv preprint arXiv:1801.07736, pp. 1–17, 2018.
[31] Y. Keneshloo, T. Shi, N. Ramakrishnan, et al. Deep reinforcement learning for sequence to sequence models. arXiv preprint arXiv:1805.09461, pp. 1–22, 2018.
[32] R.S. Sutton, A.G. Barto, et al., Reinforcement Learning: An Introduction, MIT press, 1998.
[33] X. Zhang, Y. Lecun, Text understanding from scratch, Comput. Sci. (2015).
[34] W.R. Gilks, S. Richardson, D. Spiegelhalter, Markov Chain Monte Carlo in Practice, CRC Press, 1995.
[35] Y. Zhang, Z. Gan, K. Fan, et al., Adversarial feature matching for text generation, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017, pp. 4006–4015.
[36] J. Guo, S. Lu, H. Cai, W. Zhang, Y. Yu, J. Wang, Long text generation via adversarial training with leaked information, arXiv preprint arXiv:1709.08624, pp. 1–8, 2017.
[37] S. Schleimer, D.S. Wilkerson, A. Aiken, Winnowing: local algorithms for document fingerprinting, in: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, ACM, 2003, pp. 76–85.
[38] A.Z. Broder, S.C. Glassman, M.S. Manasse, G. Zweig, Syntactic clustering of the web, Comput. Netw. ISDN Syst. 29 (8) (1997) 1157–1166.
[39] M. Steinbach, G. Karypis, V. Kumar, et al., A comparison of document clustering techniques, in: Proceedings of the KDD Workshop on Text Mining, 400, Boston, 2000, pp. 525–526.
[40] A. Ratnaparkhi, A maximum entropy model for part-of-speech tagging, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 1996.
[41] D.A. Balota, J.I. Chumbley, Are lexical decisions a good measure of lexical access? The role of word frequency in the neglected decision stage., J. Exp. Psychol. Hum. Percept. Perform. 10 (3) (1984) 340.
[42] P. Blunsom, Hidden markov models, Current opinion in structural biology 6 (3) (1996) 361–365.
[43] S. Baccianella, A. Esuli, F. Sebastiani, Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining., in: Proceedings of the LREC, 10, 2010, pp. 2200–2204.
[44] J. Huang, S. Rogers, E. Joo, Improving restaurants by extracting subtopics from yelp reviews, in: Proceedings of the iConference 2014 (Social Media Expo), 2014.
[45] D. Davidov, O. Tsur, A. Rappoport, Semi-supervised recognition of sarcastic sentences in twitter and amazon, in: Proceedings of the Fourteenth Conference on Computational Natural Language Learning, Association for Computational Linguistics, 2010, pp. 107–116.
[46] R. Socher, A. Perelygin, J. Wu, J. Chuang, C.D. Manning, A. Ng, C. Potts, Recursive deep models for semantic compositionality over a sentiment treebank, in: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, 2013, pp. 1631–1642.
[47] Y. Yao, B. Viswanath, J. Cryan, H. Zheng, B.Y. Zhao, Automated crowdturfing attacks and defenses in online review systems, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2017, pp. 1143–1158.
[48] D.M. Diab, K.M.E. Hindi, Using differential evolution for fine tuning nave Bayesian classifiers and its application for text classification, Appl. Soft Comput. 54 (2016).



Jinyin Chen received B.S. and Ph.D. degrees in 2004 and 2009, respectively, Zhejiang University of Technology, Hangzhou, China. She is currently an associate professor at the College of Information Engineering, Zhejiang University of Technology, China. Her research interests cover deep learning, intelligent computing, complex networks, and algorithm security.

Yangyang Wu is a Master student at the Institute of Information engineering, Zhejiang University of Technology. He received his bachelor degree from Ningbo Institute of Technology, Zhejiang University in 2016. His research interest covers data mining and applications, complex networks, and clustering analysis.

Chengyu Jia is a Bachelor student at the college of Information engineering, Zhejiang University of Technology. His research interests include data mining and applications, and intelligent computing.

Haibin Zheng is a Master student at the Institute of Information engineering, Zhejiang University of Technology. He received his bachelor degree from Zhejiang University of Technology in 2017. His research interest covers data mining and applications, social network data mining, and optimization.

Guohan Huang is a Bachelor student at the college of Information engineering, Zhejiang University of Technology. His research interests include data mining and applications, and intelligent computing.