Information Sciences 450 (2018) 301–315

Contents lists available at ScienceDirect

Information Sciences

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

A Generative Model for category text generation

Yang Li[a], Quan Pan[a], Suhang Wang[c], Tao Yang[a], Erik Cambria[b,*]

[a] School of Automation, Northwestern Polytechnical University, Xi'an, Shanxi 710072, PR China
[b] School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798, Singapore
[c] Department of Computer Science and Engineering, Arizona State University, Tempe, Arizona 85281, United States

ARTICLE INFO

Article history:
Received 30 September 2017
Revised 20 March 2018
Accepted 22 March 2018
Available online 26 March 2018

Keywords:
Category sentence generation
Generative adversarial networks
Generative models
Supervised learning

ABSTRACT

The neural network model has been the fulcrum of the so-called AI revolution. Although very powerful for pattern-recognition tasks, however, the model has two main drawbacks: it tends to overfit when the training dataset is small, and it is unable to accurately capture category information when the class number is large. In this paper, we combine reinforcement learning, generative adversarial networks, and recurrent neural networks to build a new model, termed category sentence generative adversarial network (CS-GAN). Not only the proposed model is able to generate category sentences that enlarge the original dataset, but also it helps improve its generalization capability during supervised training. We evaluate the performance of CS-GAN for the task of sentiment analysis. Quantitative evaluation exhibits the accuracy improvement in polarity detection on a small dataset with high category information.

1. Introduction

The success of many existing machine learning and data mining algorithms relies on large amounts of labeled data. For example, one important reason that convolutional neural networks (CNNs) have become so popular is the emergence of large-scale datasets such as ImageNet, which contains 14,197,122 manually-labeled images [33]. The majority of existing classifiers cannot perform as expected when the size of the training dataset is small. Constructing a large labeled dataset, however, is time-consuming and sometimes it requires domain knowledge. Thus, there is a gap between the importance of having a large training dataset and the difficulty in obtaining such data.

Generative models, which can generate realistic data samples, appear to be a promising tool for augmenting data size to bridge such a gap. The essential idea of generative models is to approximate the underlying data distribution by training a model to fit the training data. With the learned data distribution, generative models can generate observable data values. Thus, a massive amount of labeled data can be generated by training a generative model from a small amount of labeled data, which can be used for training the classifiers. Various generative models have been proposed in the literature, such as latent Dirichlet distribution [3], restricted Boltzmann machines [14], and generative adversarial networks (GANs) [11], which use the adversarial training idea for generating more realistic data samples. Among the existing generative models, GANs are attracting increasing attention. The core idea of GAN is to play a min-max game between a discriminator and

* Corresponding author.
E-mail addresses: liyangnpu@mail.nwpu.edu.cn (Y. Li), quanpan@nwpu.edu.cn (Q. Pan), suhang.wang@asu.edu (S. Wang), yangtao107@nwpu.edu.cn (T. Yang), erik@sentic.net (E. Cambria).
URL: http://www.sentic.net (E. Cambria)

https://doi.org/10.1016/j.ins.2018.03.050
0020-0255/

a generator, i.e., adversarial training. The discriminator tries to differentiate between real samples and artificial samples (constructed by the generator) while the generator tries to create realistic samples that can fool the discriminator (i.e., make the discriminator believe that the generated samples are real). GANs have shown an extremely powerful ability to generate artificial images and facilitated many applications. For example, an image generator based on GANs can create super-resolution [20] images from their low-resolution counterparts and an interactive image generator [50] can generate realistic images from some sketches or do the auto painting [24] with the help of GAN.

Because of the astonishing power of GAN in generating realistic images, its adoption in the context of natural language processing (NLP) for generating sentences is attracting increasing attention [17,23,48,49]. For example, Zhang et al. [49] and Semeniuta et al. [36] used GANs for text data generation and achieved state-of-the-art results. Dialogue-GAN proposed in [23] demonstrated the ability of GAN to generate realistic dialogues. However, existing works on text generation mainly focus on generating unlabeled sentences, which are not helpful for data augmentation to train better classifiers.

In this paper, we study the novel problem of generating labeled sentences with GAN for data augmentation. Because sentences are sequential data, recurrent neural networks (RNNs) are always used during generation. Also, the generator can be an agent whose target is to predict the next character based on current characters, which can be considered as a reinforcement learning (RL) process. Hence, in this paper, an ensemble of RNNs and RL is applied. In particular, we aim to tackle two challenges: (1) generating realistic sentences with GAN, given the discrete nature of text; and (2) incorporating category information in GAN to generate labeled synthetic data. To this end, we propose a novel framework termed category sentence–generative adversarial networks (CS-GAN), which not only can expand any given dataset by means of GANs and RL, but also can learn sentence structure directly with RNNs. Experiments show the effectiveness of the proposed model in the context of sentiment analysis, especially in the case of large category information. The main contributions of this work are as follows:

- We study a new and important problem of labeled sentence generation, which can be used to help train better classifiers;
- We propose a new framework CS-GAN, which stacks GAN, RNN and RL together for better sentence generation;
- We conduct extensive experiments to demonstrate the proposed framework can generate more realistic sentences with labels.

The remainder of the paper is organized as follows: Section 2 illustrates the literature of both models and tasks related to the research work; Section 3 introduces preliminaries of the proposed model; Section 4 describes CS-GAN in detail; Section 5 validates the effectiveness of the proposed model; finally, Section 6 offers concluding remarks.

2. Related works

In this section, we illustrate related works for the models we use (namely: RNN, GAN, and RL) and for sentence generation and sentiment analysis (the focus and context of this paper, respectively).

2.1. Recurrent neural networks

Because of its recurrent structure, RNN is good for sequence data processing. There is no constraint for the sequence length when applying this model, and the hidden unit is updated at every time-step. One of the early RNN models was BiRNN [35], which splits the neurons of regular RNN into two directions: one for the forward computation and one for backward computation. Today, the most successful RNN model is the long short-term memory (LSTM) [16] network, where the gates in each neuron help the model predict the sequence data better based on contextual tokens. Many more models based on LSTM have been proposed recently, e.g., bidirectional LSTM [12], gated recurrent neural tensor network (GRNTN) [42] etc. These works not only help the sequence data generation but also make the model more flexible when faced with the variety of sequence data.

Many works employed LSTM for sentence generation, either directly [40] or as an embedded module [48,49]. Some works [9] use LSTM for machine translation on the basis of sentence generation, other works deploy this model for end-to-end speech recognition based on sequence generation [38,44]. All of these models leverage the so-called teacher-forcing algorithm [45], which teaches the generation using near future tokens. This algorithm was later improved by Alex et al. [19], who introduced the professor-forcing algorithm, which outperforms teacher-forcing methods in text generation by using dynamic RNNs and by teaching the generator a large range of existing tokens. Prior information, e.g., sentence sentiment, can be added to this model during sequence data generation, which makes the generation more flexible. Some works added category information, which aids category sentence generation, e.g.,[10] combined conditional LSTM and category information for sentence classification in semi-supervised learning. Our work also employs LSTM and prior category information for sentence generation but our goal is to use the generated category sentence to improve the generalization of supervised learning.

2.2. Generative adversarial networks

Recently, GAN [11] has become very popular in the context of computing vision. Many models are based on GAN to generate images from a predefined distribution. In such models, the discriminator has the goal to distinguish between artificial

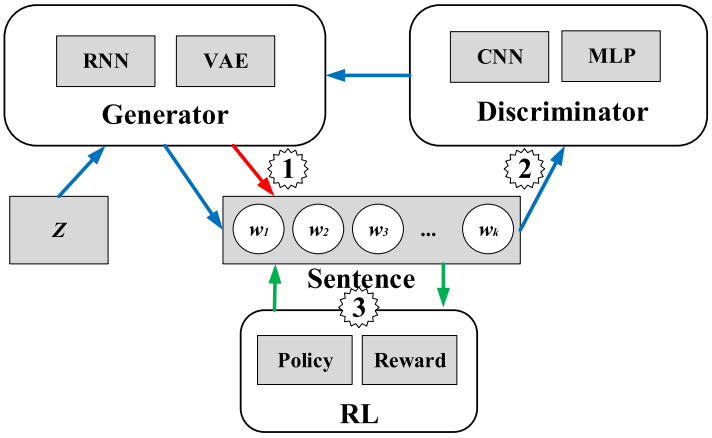


Fig. 1. Sentence generation models in previous works. The different color arrow lines stand for the information streams in different text generation models, and the spiny round with number denotes the class number. Spiny round 1 together with the red arrow stands for the first class, spiny round 2 together with blue arrows denotes the second class, and spiny round 3 together with the green arrows is the third class. w_k is the tokens in the sentence, z is the prior distribution in GANs. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

images (created by the generator) and real ones. The zero-sum game between the generator and discriminator helps improve their respective abilities step by step. Because GAN is a bit unstable when training the network, some methods were proposed to avoid collapse during training, e.g., WGAN [1], LossSensitiveGAN [32], Improved GAN [34]. Some works proposed to integrate extra information into GAN, e.g., Info-GAN [8], Cat-GAN [39], and other works [39] used GAN in semi-supervised learning, in which GAN generates samples for training the classifier.

There are some obstacles in applying GAN to NLP [11], e.g., the discrete space of words that cannot be differentiated in mathematics. To this end, works like Seq-GAN [48] and Dialogue-GAN [23] applied RL for text generation by using softmax over continue values for character selection. Controllable text generation [17] applies the variable auto-encoder (VAE) together with controllable information to generate category sentences. Zhang et al. [49] and Semeniuta et al. [36] used GANs for text data generation and achieved state-of-the-art results.

2.3. Reinforcement learning

It is natural to consider sentence generation as the decision-making process of an agent that takes actions (next character selection) based on a goal-oriented policy aimed at achieving best long-term rewards. There are several models of RL [41], some of which were applied to sentence generation, e.g., actor-critic algorithm [2] and deep q-network [13]. In SeqGAN [48], in particular, the Monte Carlo method is used to search for next tokens. This is also applied to dialogue generation [23] and neural network decoder with specific properties [22].

Recently, researchers have been looking for relations between RL and GAN [15,30]. In particular, Ho et al. [15] found a connection between GAN and inverse RL [29], believing that a transformation can be made between the generative adversarial imitation learning and reverse RL using the entropy-regularized term.

2.4. Sentence generation

Sentence generation consists in producing natural language from a computational representation of information. There are some seminal works on generating good sentences using GANs (Fig. 1). Most of them consider sentence generation as a process of character prediction and use RNN for feature extraction from time series data [40,48,49]. There are also some works treating sentence generation as the encoder-decoder problem, which aim to minimize the loss between the source data and the target data. Recently, VAE achieved state-of-art results in sentence generation [17,36].

Because all of these models are used for generating the sentence directly, we can put them into the same class which is represented by the spiny round with number one (Fig. 1). The information stream in those models is represented by the red arrow line nearby. Besides the generator (RNN, VAE, etc.), which produces synthetic sentences from the known distribution z, a discriminator is introduced to evaluate the generated data (and, hence, help the generator perform better). A zero-sum game is played by these two roles, which improves the quality of the generation step by step under the framework of GAN [17,48,49]. Thus, models with generator and discriminator can be set as the second class in which the information stream is represented by the blue arrow lines nearby in Fig. 1. Despite the astonishing success of GANs in image generation, generating sentences and documents using GANs is seldom studied and remains a challenging problem. The main difficulty of generating texts using GANs lies in the discrete nature of texts, which limits the differential propagation in GANs. Unlike image pixels (which are represented as the real number within a certain range), in fact, words or tokens of documents are

discrete and are usually represented as one-hot coding. This can be solved temporarily using the softmax function during token selection.

Sentence generation can also be treated as a decision-making process, which is sorted as the third class, where an agent selects characters from a dataset based on a policy that leads to best long-term rewards [22,48]. This model information stream is represented by the green arrow lines in Fig. 1.

2.5. Sentiment analysis

In recent years, sentiment analysis has become increasingly popular for processing social media data on online communities, blogs, wikis, microblogging platforms, and other online collaborative media. Sentiment analysis is a branch of affective computing research [31] that aims to classify text – but sometimes also audio and video – into either positive or negative – but sometimes also neutral [6]. Most of the literature is on English language but recently an increasing number of publications is tackling the multilinguality issue [25].

While most works approach it as a simple categorization problem, sentiment analysis is actually a suitcase research problem [4] that requires tackling many NLP tasks, including word polarity disambiguation [46], concept extraction [5], subjectivity detection [7], personality recognition [27], and aspect extraction [26].

Sentiment analysis has raised growing interest both within the scientific community, leading to many exciting open challenges, as well as in the business world, due to the remarkable benefits to be had from marketing and financial forecasting [47].

3. Preliminaries

The common way to achieve data augmentation is to generate labeled sentences which can capture the true data distribution. In order to capture the feature of the existing data distribution, we divide the generation process in two steps: adding category information into the model and forcing the model to generate category sentences accordingly. In this section, we outline the basic sentence generation models.

It is difficult to generate natural sentences when the dictionary volume is large; because of the large searching space, selecting the next token is time-consuming and precision-compromising. To limit the action space (dictionary volume), the model is built at the character level. RNN, described in next section, is used as the basic sentence generator. Then, we describe how we employed GAN and RL.

3.1. Recurrent neural networks

The most common way for sentence generation is using RNN, which has achieved impressive results [40]. All of these methods are teacher-forcing models [45] and they predict the next token in a stream of text via supervised learning. There are different compositions about the input and output numbers in RNN which enable it to be designed flexibly according to different applications. In this paper, we apply RNN as the character predictor with one output and sequence of input and make the best prediction of $p(x_{t+1}|y_t)$, where x_{t+1} is the predicted character and y_t is the current state.

3.2. Generative adversarial networks

To the best of our knowledge, there are very few works on text analysis using GANs. In this paper, we take advantage of GAN by applying RNN as the generator and the generated sentence is scored by the discriminator. A recent work [17] uses VAE to solve the problem of the high variance in model training, and add the controllable information in the VAE. Unlike that work, we encode the text stream during the last layer of the RNN and we generate the sentence from the original data directly. After generation, the real sentences and the generated sentences are fed into the discriminator separately. Similar to vanilla GAN models, it is a zero-game theory between the generator and the discriminator.

3.3. Reinforcement learning

Inspired by the model of SeqGAN [48], sentence generation can also be regarded as a game-playing process, in which the agent chooses next character based on the current state to achieve long-term rewards while the discriminator aims to achieve immediate rewards. The main challenge lies in the policy gradient updating which has to be performed after the sentence is generated, because we can only get the reward from the discriminator who gives the score over the whole sentence. SeqGAN [48] addresses this by using Monte Carlo searching with rollout technique to get the reward from the generated tokens. The drawback of this method is that it is time-consuming. Because the action space is reduced, in this work the sentence generation time is shortened at the same time.

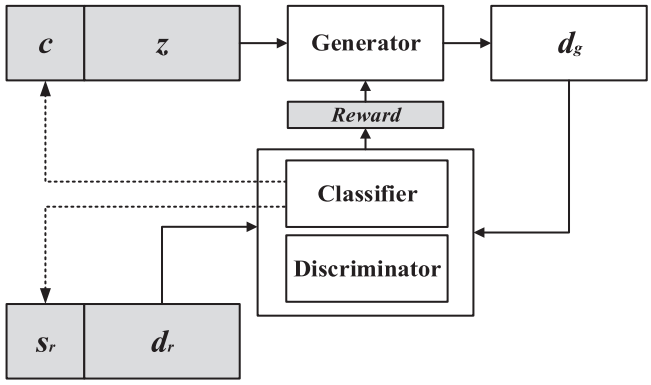


Fig. 2. Structure of CS-GAN: c is the structured category information, z is the input distribution, d_g is the output sentence from the generator, s_r and d_r are the real label and sentence respectively, the dash arrows stand for the constraints.

4. CS-GAN model

In this section, the proposed model (CS-GAN) is introduced. Recently, models like CAT-GAN [39], Info-GAN [8] have been trying to integrate category information into the generated data. Some models join the label as the input [17,39], some others regard the label as the target that needs to be predicted [8]. To make the sentence generation controllable, the label information is used as the input. Inspired by the work of Hu and Yang et al. [17], the controllable information c and the sentence distribution z are concatenated together to be the prior information. In our framework, there are two parts which are generator and descriptors respectively playing the min-max game. As we have described before, RNNs and RL are applied in the generator. As to the descriptor which contains classifier and discriminator, and these two parts are for the labeled synthetic sentences generation.

4.1. Generator

To avoid gradient vanish, in this paper we use LSTM as a generator. We use a classifier to ensure that the generated sentence contains the label information.

As we have described earlier, the category information is added at each generating step. The prior category vector concatenates word embedding at each time-step, which is widely applied in [37]. Together with the latent variable z, the generator that using the LSTM can be depicted by the following equations:

$$f_t = \sigma\left(W_f[x_t; z; c] + U_f h_{t-1} + b_f\right) \tag{1}$$

$$i_t = \sigma\left(W_i[x_t; z; c] + U_i h_{t-1} + b_i\right) \tag{2}$$

$$o_t = \sigma\left(W_o[x_t; z; c] + U_o h_{t-1} + b_o\right) \tag{3}$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes \sigma\left(W_c[x_t; z; c] + U_c h_{t-1} + b_c\right) \tag{4}$$

$$h_t = o_t \otimes relu(c_t) \tag{5}$$

The above equations are the same as the ones of vanilla LSTM models, except for the concatenation term of current word embedding x_t, latent variable z, and the controllable information c.

Based on the generated tokens, the agent (the generator) takes action a (the token set) and then the descriptors (discriminator/classifier) return rewards about the current status. The structure of CS-GAN is shown in Fig. 2.

In this model description, we will present the generator and the descriptors (discriminator/classifier) respectively, where the generator is to generate the category synthetic sentences, and the discriminator and the classifier are to evaluate them in sentence validity and category accuracy respectively.

The generator selects next token d_g^t based on the current states $d_g^{<t}$. Same with the case in SeqGAN [48], there is also no the immediate reward after the action. To address that problem, the rollout technique is deployed, and the generator parameter is G_γ which could be same with the G_{θ_g} during the rollout process. Then the rollout sentences are fed into the descriptors to get the rewards.

Generally, the generator tries to gain the maximum expect reward as follows:

$$J(\theta_g) = \mathbb{E}[R_T|w_0, c, \theta_g] = \sum_{\hat{d}_g \in W} G_{\theta_g}(\hat{d}_g|w_0, c) Q^{G_{\theta_g}}_{<D_{\theta_d}, D_{\theta_c}>}(w_0, \hat{d}_g), \tag{6}$$

where R_T is the reward of the rollout sentence, w_0 is the start state, c is the controllable information, θ_g, θ_d, θ_c are the parameters of the generator, discriminator and the classifier respectively, $\hat{d}_g$ is the generated token, D_{θ_g} is the probability of a sentence is real, D_{θ_d} is the probability of the sentence in the right category, and G_{θ_g} is the generated results based on the latent variable (w_0, c) through RNN-LSTM which can be depicted as follows:

$$G_{\theta_g}(\hat{d}_g|w_0, c) = p_G(\hat{d}_g|w_0, z, c) = \prod_t p(\hat{d}_g^t|d_g^{<t}, z, c), \tag{7}$$

where current state $w_0 = d_g^{<t}$ is the generated tokens before t, $Q^{G_{\theta_g}}_{<D_{\theta_d}, D_{\theta_c}>}(w_0, \hat{d}_g)$ is the action-value function during the selection process. Unlike SeqGAN, CS-GAN contains two roles in the status judgments, which help the generation to be more adaptive. Meanwhile, the model is applied at character level, which restrains the size of the action space to a limited number. To ensure the existence of the gradient in the sequence deciding process, softmax function is used when mapping the real values to the tokens, so we have

$$\hat{d}_t \sim softmax(o_t/\tau), \tag{8}$$

where o_t is the logistic vector from $G_{\theta_g}(\hat{d}_g^t, |w_0, c) Q^{G_{\theta_g}}_{<D_{\theta_d}, D_{\theta_c}>}(w_0, \hat{d}_g)$, and τ is the temperature which is normally 1. As mentioned earlier, the reward comes from the discriminator and classifier simultaneously. To balance these two descriptors when estimating the action-value function, one way is to resemble F-measure:

$$Q^{G_{\theta_g}}_{<D_{\theta_d}, D_{\theta_c}>}(w_0, \hat{d}_g) = \frac{2 D_{\theta_d} D_{\theta_c}}{D_{\theta_d} + D_{\theta_c}} \tag{9}$$

where the last layer of the discriminator and the classifier is the softmax function, which normalizes these two output values equally.

According to Eq. (6), the update in the generator parameters with respect to θ_g can be derived as:

$$\theta_g \leftarrow \theta_g + \alpha_t \bigtriangledown_{\theta_g} J(\theta_g), \tag{10}$$

where α_t denotes the learning rate at step t.

4.2. Descriptors

For discriminator and classifier, CNNs are utilized. In particular, the structure of both classifier and discriminator are the same as in [18]. The classifier not only distinguishes sentence categories, but also leads the category sentence generation. Thus, we have the following equations for the classifier optimization:

$$L_C = Loss(< C(d_r; \theta_c), s_r >) + Loss(< C(G(z; \theta_g); \theta_c), \varphi(c) >) \tag{11}$$

where $C(\cdot)$ is the logistic loss of the classifier, φ is a linear function of the controllable label generation, θ_c and θ_g are the parameters of the classifier and the RNN model. The first term in the right hand side is the loss from the true sentence, and the second term is the loss from the generated sentence.

As to the discriminator, following the characteristic of GANs, the discriminator can be described as follows:

$$\begin{aligned} L_D &= H_{\mathbb{E}_{d_g \sim (c,z)}} + H_{\mathbb{E}_{d_r \sim data}} \\ H_{\mathbb{E}_{d_g \sim (c,z)}} &= \mathbb{E}_{d_g \sim (c,z)}[-log(1 - D(G(\varphi(c), z, \theta_g)), \theta_d)] \\ H_{\mathbb{E}_{d_r \sim data}} &= \mathbb{E}_{d_r \sim data}[-log(D(s_r, d_r, \theta_d))] \end{aligned} \tag{12}$$

where d_g and d_r are the generated tokens and real tokens respectively, c is the controllable information, s_r is the real data label, θ_g and θ_d are the parameters of the generator and the discriminator, φ is the linear function, z is the distribution of the generated data.

As the descriptors are strong, they can distinguish the generated data easily, which leads to poor behavior of the generator. To balance the two descriptors and the generator, we have to pre-train a sound generator at first using an RNN-based model. During the training, the uniform distribution is adopted to generate the first character, and then the sentence is generated token by token. This model is depicted in Algorithm 1. To avoid high discrimination results based on poor generation, the discriminator will stop updating when the discriminator accuracy Dis_{acc} is higher than 0.9.

Algorithm 1 The algorithm of CS-GAN.

Require: generator G_{θ_g}; roll-out G_γ;discriminator D_{θ_d}; Classifier D_{θ_c}; Dataset $W = d_{1:t}$

Initialize the parameters of G_{θ_g}, D_{θ_d} and D_{θ_c}
Use the positive samples W to pre-train the generator
$\gamma \leftarrow \theta_g$
Generate the negative samples using G_γ for training D_{θ_d} and D_{θ_c}
Pre-train discriminator and classifier
repeat
  for g-steps **do**
    Generate a sequence $d_{1:t} \sim G_{\theta_g}$
    for t in 1:T **do**
      Compute $Q^{G_{\theta_g}}_{<D_{\theta_g}, D_{\theta_d}>}$ using Eq. (9) with rollout samples
    end for
    Update parameters of generator in Eq. (10).
  end for
  for d-steps **do**
    if $Dis_{acc} < 0.9$ **then**
      Update the parameters of discriminator with the generated sentences and real sentences using Equation 12.
    end if
    for classifier-epochs **do**
      Train the classifier with generated labeled sentences and the real labeled sentences using Equation 11.
    end for
  end for
  $\gamma \leftarrow \theta_g$
until model convergence

5. Experiments

It is hard to make a good evaluation for text generation because there is no objective way to assess whether an artificial sentence is more plausible or realistic than another. To this end, some works like SeqGAN [48] use BLEU score, which is designed to evaluate machine translation quality. Zhang et al. [49] use Jensen–Shannon divergence between two distributions, while works in [10,19,48] apply the negative log-likelihood (NLL) to measure the prediction. Because one task of CS-GAN is to generate text, NLL is used as the one measurement in this work. Next, to validate the proposed model for category sentence generation, sentiment analysis on a different dataset is performed, where the polarity classification accuracy is the main measurement.

5.1. Dataset

We extract the sub-datasets from Amazon review dataset [28], Yelp review dataset,[1] Stanford sentiment tree bank dataset (SST),[2] NEWS dataset that crawled from NYTimes, Reuters and USAToday [43], and Emotion dataset[3] which is provided by CrowdFlower.[4] To make comparisons and the evaluate the model over small datasets, a small sub-dataset is selected from those datasets, respectively (except for the sub-datasets from SST). (Table 1)

In the Amazon review dataset, there are two categories and the maximum length is no more than 120 characters, the small sub-dataset which is named Amazon-5000 contains 5000 training sentences, 2000 development sentences, and 2000 test sentences, and the large sub-dataset Amazon-30000 has 30,000 training sentences, 10,000 development sentences, and 10,000 test sentences. As to the sub-dataset from the NEWS and Emotions, the maximum sentence length is 154 and 150 respectively, and NEWS-15000 together with Emotions-15000 have the same size for the training set, development set and the test set. The suffixes of “Amazon”, “NEWS” and “Emotions” represent the size of the training set.

For the validation of the sentence length in sentiment analysis, we use the sub-datasets from Yelp and SST. Unlike the dataset mentioned before, we extract four sub-datasets by the sentence length from SST and Yelp respectively, which are Yelp-1000, Yelp-800, Yelp-500, Yelp-200 and SST-120, SST-100, SST-80, SST-60 respectively (suffixes denoting the maximum length of a sentence at character level). The details of those sub-datasets are shown in Table 3.

All of these experiments are running on the GTX-TITAN graphics card which is equipped with a 12GB GPU memory.

[1] http://yelp.com/dataset.
[2] http://nlp.stanford.edu/sentiment/treebank.html.
[3] http://data.world/crowdflower/sentiment-analysis-in-text.
[4] http://crowdflower.com.

Table 1
The details of the sub-datasets.

Dataset name	Classes	Train	Dev	Test	Max Len
Amazon-5000	2	5000	2000	2000	120
Amazon-30000	2	30,000	10,000	10,000	120
NEWS-15000	7	15,000	5000	5000	154
Emotion-15000	13	15,000	5000	5000	150
Yelp-1000	5	15,000	5000	5000	1000
Yelp-800	5	15,000	5000	5000	800
Yelp-500	5	15,000	5000	5000	500
Yelp-200	5	15,000	5000	5000	200
SST-120	2	4440	553	1176	120
SST-100	2	3475	421	906	100
SST-80	2	2452	291	634	80
SST-60	2	1556	167	397	60

5.2. Compared methods

To validate the different parts contributing to category sentence generation, we unstack the components of CS-GAN by removing one item at the time. In the descriptor, the CNN structure is taken from [18], which is a one-layer convolution on top of the character vectors, and max-pooling is applied to get a dense vector for the final fully-connected layer.

5.2.1. CS-GAN without RL

To test the contribution of RL, we remove it from CS-GAN to check how results change in absence of RL. Without RL, the generator deciding the next token mainly depends on the current status. All of the roles in CS-GAN are reminded, except for the process of Monte Carlo search and the updating policy, so the rule of updating the generator parameters changes as follows:

$$J(\theta_g) = \sum_{\hat{d}_g \in W} G_{\theta_g}(\hat{d}_g | w_0, c) \tag{13}$$

As in the original CS-GAN, the discriminator measures the generated sentence by grading the score to the whole sentence, while the classifier and the discriminator are all the descriptive roles whose functions are to map signals to features.

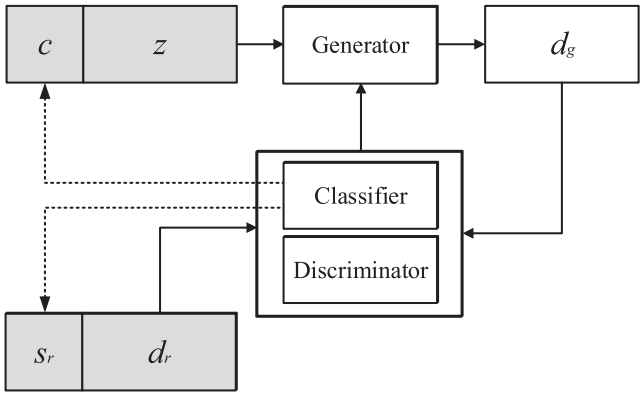
Similarly, the classifier and the discriminator are trained according to Eqs. (11) and (12), respectively. This model is shown in Fig. 3(a) and the algorithm is depicted in Algorithm 2.

Algorithm 2 The algorithm of CS-GAN without RL.

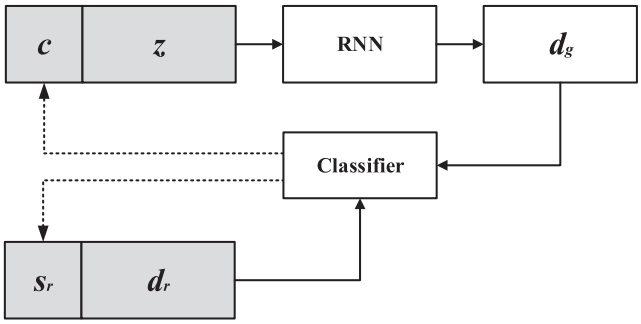
Require: generator G_{θ_g};discriminator D_{θ_d}; Classifier D_{θ_c}; Dataset $W = d_{1:t}$
1: Initialize the parameters of G_{θ_g}, D_{θ_d} and D_{θ_c}
2: Use real sentences W to pre-train the generator
3: Pre-train the discriminator and classifier with real labeled sentences
4: **repeat**
5: **for** g-steps **do**
6: Generate a sequence $d_{1:t} \sim G_{\theta_g}$
7: Update parameters of generator in Eq. (13).
8: **end for**
9: **for** d-steps **do**
10: **if** $Dis_{acc} < 0.9$ **then**
11: Update the parameters of discriminator with the generated sentences and real sentences using Eq. (12).
12: **end if**
13: **for** classifier-epochs **do**
14: Train the classifier with generated labeled sentences and the real labeled sentences using Eq. (11).
15: **end for**
16: **end for**
17: **until** model convergence

5.2.2. CS-GAN without RL & GAN

Next, we remove GAN from CS-GAN without RL. This model is depicted in Fig. 3(b), which can be treated as the RNN model. The left roles are the generator and the classifier, and the discriminator is removed along with GAN. The algorithm is depicted in Algorithm 3.



(a) The structure of CS-GAN without RL, where d_g is the output sentence from the generator, s_r and d_r are the real label and sentence respectively. The dash arrows stand for the constraints.



(b) The structure of CS-GAN without RL & GAN, where d_g is the output tokens from the RNN, s_r and d_r are the real label and tokens respectively. The dash arrows stand for the constraints.

Fig. 3. The structures of the compared models.

Algorithm 3 The algorithm of CS-GAN without RL&GAN.

Require: generator G_{θ_g}; Classifier D_{θ_c}; Dataset $W = d_{1:t}$
Initialize the parameters of G_{θ_g}, and D_{θ_c}
Use the real sentences W to pre-train the generator
Pre-train the classifier with the real labeled sentences
repeat
  for g-steps **do**
    Generate a sequence $d_{1:t} \sim G_{\theta_g}$
    Update parameters of generator in Eq. (13).
  end for
  for classifier-epochs **do**
    Train the classifier with generated labeled sentences and the real labeled sentences using Eq. (11).
  end for
until model convergence

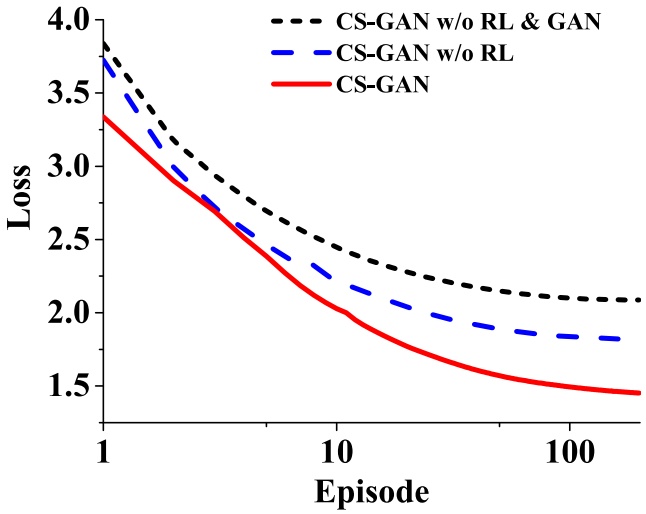


Fig. 4. The results of NLL from model CS-GAN, CS-GAN without RL and CS-GAN without RL & GAN in text data generation.

5.3. Experiment results

The performance of CS-GAN in sentence generation is described in Section 5.3.1, CS-GAN's sentence feature capturing is illustrated in Section 5.3.2, and category sentence in supervised learning generalization is depicted Section 5.3.3.

5.3.1. Text generation

To validate the proposed model in text generation, NLL is used as the measurement over the data of Amazon-5000. Results are shown in Fig. 4.

From Fig. 4, we can see that CS-GAN has the best NLL convergence, CS-GAN without RL is in the second place, and CS-GAN without RL and GAN performs the worst. In other words, both GAN and RL contribute to improving CS-GAN's performance in sentence generation.

5.3.2. Feature representation

In order to examine how well the features of the generated sentences have been captured, feature consistency among the synthetic and real sentences is tested [49]. We extract the features from the top layer of the classifier, which is represented as a 1024-dimensional vector f. Then, we use empirical expectation to represent the outer consistency between two distributions, and covariance of the features over the real sentences and synthetic sentences to represent the inner consistency. These two set values are scattered in Fig. 5. The covariance is the 1024 features against the first feature (i.e., $Cov(f_i, f_1)$) in each dataset.

As shown in Fig. 5, all models produce consistency features with the real sentences, which means that it is easy to achieve the inner consistency in one distribution. From the expectation, however, we can see that the features from CS-GAN fit the real sentences best. Features from the other two models have some inconformity with those in the real sentences, especially in the model without RL and GAN. From this perspective, we can see that it is hard to bridge the gap between the two distributions, but CS-GAN can achieve both consistencies well.

5.3.3. Text augmentation for classification

A good generation can provide a solid training set which guarantees text augmentation in a classification task, and this releases the pressure from the lack of labeled data. The selected examples of the generated category sentences by CS-GAN are shown in Table 2.

From Table 2, we can see that CS-GAN captures the category information. Though the generated sentences are short, they are related to their label information. To qualitatively analyze the performance of the generalization with different generation numbers in supervised learning, the number of the generated sentence is set as follows $N_{\text{generated}} = \beta N_{\text{labeled}}$, where β is generation ratio which is set to 0.025, 0.5, 1.0 respectively. Here N_{labeled} denotes the number of labeled training sentences, and $N_{\text{generated}}$ is the number of the generated sentences. The evaluation results are depicted in Fig. 6.

From the figure, we can see that when β is small (i.e., when there are few generated labeled sentences in the training process), the classifier will overfit easily. As beta increases (that is, as more synthetic data are generated to help training), the generalization ability of the model also increases, which suggests that the proposed framework can generate realistic sentences that are able to help a classifier to learn better.

To further test the effectiveness of the model in supervised learning, the datasets Amazon-5000, Amazon-30000, NEWS-15000, and Emotion-15000 are used, and the generation ratio β is set to 1.0. The classification results show in Table 3.

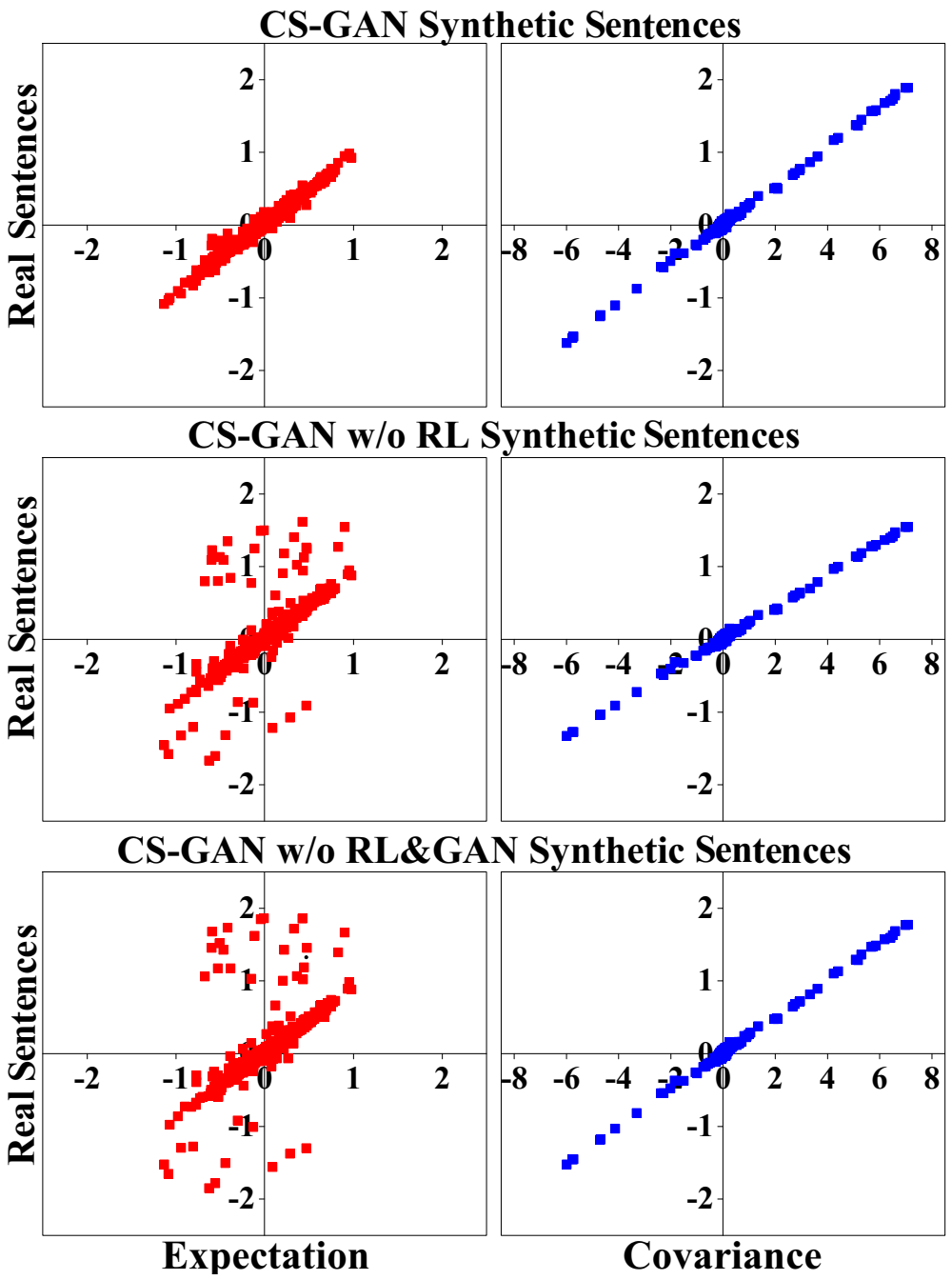


Fig. 5. Left: the scatter plot of feature expectations from real sentences against the synthetic sentences generated by CS-GAN, CS-GAN without RL and CS-GAN without RL & GAN, respectively. Right: The scatter plot of feature covariances from the real sentences against the synthetic sentences that generated by CS-GAN, CS-GAN without RL and CS-GAN without RL & GAN, respectively.

Except for the first row (which is the supervised learning by CNN), all rows are the result of CS-GAN and the compared models that are related to CS-GAN.

The table shows that in the smaller original labeled dataset with two categories, like Amazon-5000, CS-GAN achieves the best accuracy, then comes to CS-GAN without RL, the last one is CS-GAN without RL & GAN. It is just reversed in the case of Amazon-30000, and the difference in those models is slight. From this point, we can see that CS-GAN performs well with a small labeled dataset with little category information. When there is a large labeled dataset with little category information, the little category information will lead to a perfect discriminator, which decreases the generated sentence quality and, hence, produces a worse data augmentation. That is why CS-GAN does not perform well on Amazon-30000. In the

Table 2
The examples of the generated sentences.

SST	POSITIVE This is really lead movie. You take the one better.	NEGATIVE As film and day waste. This movie is mess.
Emotion	LOVE Love the warm Oh thank you my dear	EMPTY Is telling a story? Read the book boring
	RELIEF Read the book Thanks	ANGER I hate the hat Jest the boy
	SURPRISE Weather today is a gift to us I hate that happend	NEUTRAL Have a tea please. So the heat is here
	HAPPINESS haha Enjoy the game	SADNESS I hate it No body is at eating
	FUN Yeah Have the fun	ENTHUSIAMSM Love the health Love the eat
NEWS	SPORT Ball center with two teams In a race, the man wade in river	BUSINESS There is a flat race on euro It is easy a pricing ran in tea
	ENTERTAINMENT We see it ease a strong man Not a keen actor	US U.S. tears a ratar in the u.s. Japan for a real anneal stress
	WORLD Hear the trade war end Europe end this under the stress	HEALTH Done in a dead spar Argue a 1 star new rush
	SCI-TEC Tale in fick nart ear at bet Nets play and in Brate	

Table 3
The classification results on the different size dataset.

Model name	Amazon-5000	Amazon-30000
CNN	84.83%	89.55%
CS-GAN w/o RL&GAN	85.60%	**89.67%**
CS-GAN w/o RL	86.18%	89.54%
CS-GAN	**86.43%**	89.34%
Model name	Emotion-15000	NEWS-15000
CNN	40.75%	72.08%
CS-GAN w/o RL&GAN	39.32%	72.31%
CS-GAN w/o RL	40.14%	72.09%
CS-GAN	**41.52%**	**74.33%**

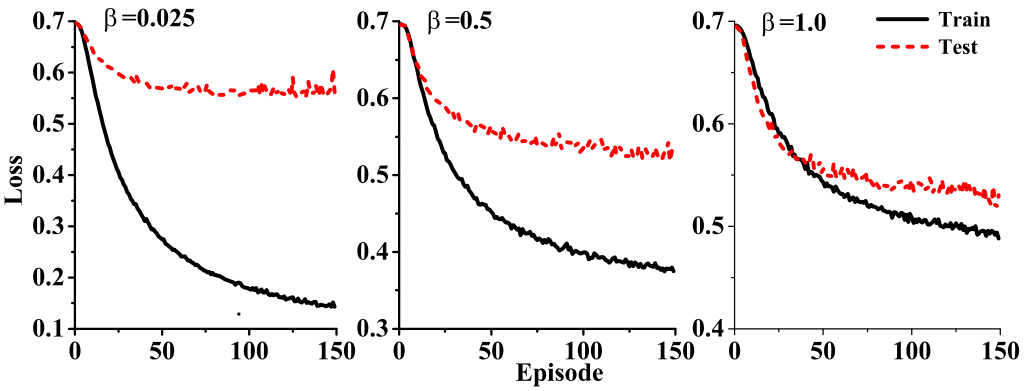


Fig. 6. The results of the losses from CS-GAN using different numbers of generated data as training data, and β is the generation ratio.

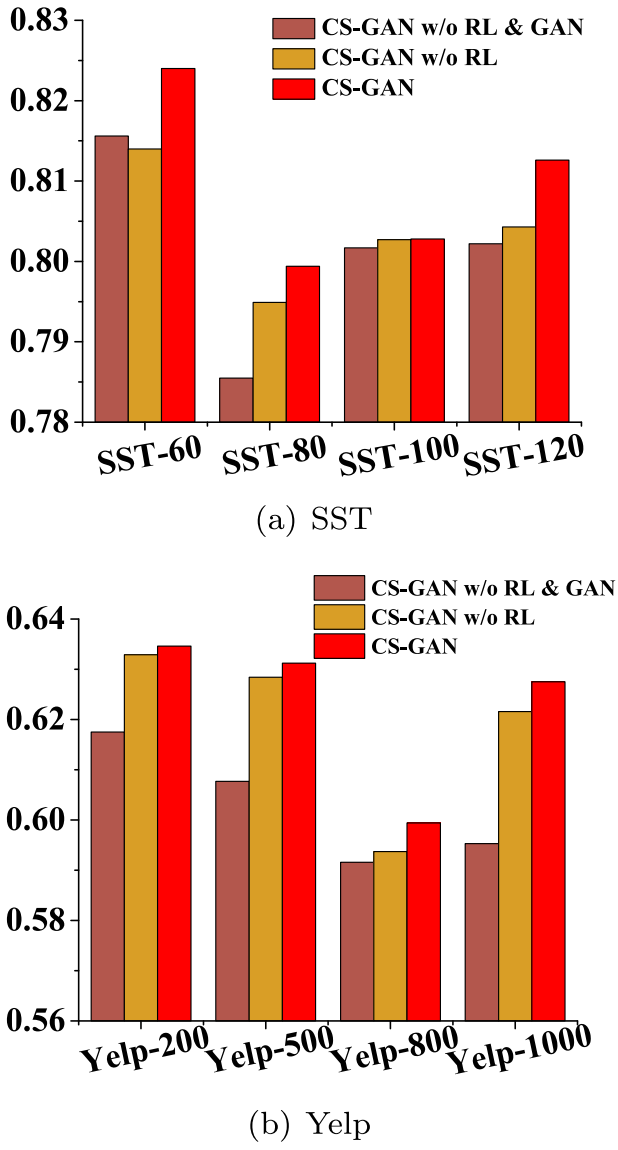


Fig. 7. The accuracy of the sentiment analysis in the sub-datasets of SST and Yelp using the proposed model.

multi-categories datasets, however, the model performs better than the compared models in the large sub-datasets, which is reflected in the results of NEWS-15000 and Emotions-15000. In those cases, the large category information makes a less perfect discriminator, which improves the quality of the generated sentences and, hence, leads to a better data augmentation. Thus, when category information is little, CS-GAN performs well with a small labeled dataset in supervised learning (which remedies the lack of the data), but when the original dataset is large, the strength of CS-GAN is weakened. In any case, the proposed model doubled the sentences number which helps the model achieve the best results when category information is high.

In the previous experiments, the maximum sentence length of the datasets is fixed. Therefore, to test the performance of the proposed model in different maximum length sentences, the datasets extracted from SST and Yelp on the basis of the sentence length are used. The results of the classification are shown in Fig. 7. We can see that, in general, the shorter the sentence, the better the results. When sentence length increases, average accuracy is lower but the full CS-GAN always outperforms its downgraded versions in both datasets. This is mainly caused by the fact that the sentences generated by CS-GAN are short, as it can be seen in Table 2. This is a common sign in sequential generative models [21]: the optimization for the likelihood objective function prefers giving the “safe” response when predicting the phrase. Hence, it is hard for those variants to get a good sentence generation when faced with a long sentence, and that is why the classification results decrease in all of the three variants with the increase of sentence length.

6. Conclusion

In this paper, the proposed a new model for text generation, termed CS-GAN, which can capture sentence structure well and include category information useful for supervised learning. CS-GAN is an ensemble of RNN, GAN, and RL. In the experiments, we showed how each of these models contributes to the performance of the proposed model. The proposed model performs well in supervised learning on the basis of the generated sentences, especially in the multi-categories datasets. However, this advantage can be weakened by a large amount of data when there is little category information (two classes). We also validated the proposed model on the task of sentiment analysis, where CS-GAN showed the best performance with different sentence lengths, especially for a small labeled dataset with short sentence length.

In the future, we plan to further explore sentence feature extraction and model a disentangled representation based on different properties using the generative models.

Acknowledgments

This work was supported by the Young Scientists Fund of the National Natural Science Foundation of China [Grant No.61402373] and Aeronautical Science Foundation of China Key Laboratory Project [Grant No.20155553036].

References

[1] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein gan, arXiv preprint arXiv:1701.07875 (2017).
[2] D. Bahdanau, P. Brakel, K. Xu, A. Goyal, R. Lowe, J. Pineau, A. Courville, Y. Bengio, An actor-critic algorithm for sequence prediction, arXiv preprint arXiv:1607.07086 (2016).
[3] D.M. Blei, A.Y. Ng, M.I. Jordan, Latent dirichlet allocation, J. Mach. Learn. Res. 3 (2003) 993–1022.
[4] E. Cambria, S. Poria, A. Gelbukh, M. Thelwall, Sentiment analysis is a big suitcase, IEEE Intell. Syst. 32 (6) (2017) 74–80.
[5] E. Cambria, S. Poria, D. Hazarika, K. Kwok, SenticNet 5: discovering conceptual primitives for sentiment analysis by means of context embeddings, AAAI, 2018.
[6] I. Chaturvedi, E. Cambria, R. Welsch, F. Herrera, Distinguishing between facts and opinions for sentiment analysis: survey and challenges, Inf. Fusion 44 (2018) 65–77.
[7] I. Chaturvedi, E. Ragusa, P. Gastaldo, R. Zunino, E. Cambria, Bayesian network based extreme learning machine for subjectivity detection, J. Franklin Inst. 355 (4) (2018) 1780–1797.
[8] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, P. Abbeel, Infogan: Interpretable representation learning by information maximizing generative adversarial nets, in: Advances in Neural Information Processing Systems, 2016, pp. 2172–2180.
[9] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using rnn encoder-decoder for statistical machine translation, arXiv preprint arXiv:1406.1078 (2014).
[10] A.M. Dai, Q.V. Le, Semi-supervised sequence learning, in: Advances in Neural Information Processing Systems, 2015, pp. 3079–3087.
[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in neural information processing systems, 2014, pp. 2672–2680.
[12] A. Graves, S. Fernández, J. Schmidhuber, Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition, Artificial Neural Networks: Formal Models and Their Applications–ICANN 2005, 2005. 753–753
[13] H. Guo, Generating text with deep reinforcement learning, arXiv preprint arXiv:1510.09202 (2015).
[14] G. Hinton, A practical guide to training restricted boltzmann machines, Momentum 9(1) (2010) 926.
[15] J. Ho, S. Ermon, Generative adversarial imitation learning, in: Advances in Neural Information Processing Systems, 2016, pp. 4565–4573.
[16] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9(8) (1997) 1735–1780.
[17] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, E.P. Xing, Controllable text generation, arXiv preprint arXiv:1703.00955 (2017).
[18] Y. Kim, Convolutional neural networks for sentence classification, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1746–1751.
[19] A.M. Lamb, A.G.A.P. GOYAL, Y. Zhang, S. Zhang, A.C. Courville, Y. Bengio, Professor forcing: a new algorithm for training recurrent networks, in: Advances In Neural Information Processing Systems, 2016, pp. 4601–4609.
[20] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al., Photo-realistic single image super-resolution using a generative adversarial network (2017) 4681–4690.
[21] J. Li, M. Galley, C. Brockett, J. Gao, B. Dolan, A diversity-promoting objective function for neural conversation models, in: Proceedings of NAACL-HLT, 2016, pp. 110–119.
[22] J. Li, W. Monroe, D. Jurafsky, Learning to decode for future success, arXiv preprint arXiv:1701.06549 (2017a).
[23] J. Li, W. Monroe, T. Shi, A. Ritter, D. Jurafsky, Adversarial learning for neural dialogue generation, arXiv preprint arXiv:1701.06547 (2017b).
[24] Y. Liu, Z. Qin, Z. Luo, H. Wang, Auto-painter: cartoon image generation from sketch by using conditional generative adversarial networks, arXiv preprint arXiv:1705.01908 (2017).
[25] S.L. Lo, E. Cambria, R. Chiong, D. Cornforth, Multilingual sentiment analysis: from formal to informal and scarce resource languages, Artif. Intell. Rev. 48 (4) (2017) 499–527.
[26] Y. Ma, H. Peng, E. Cambria, Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive LSTM, AAAI, 2018.
[27] N. Majumder, S. Poria, A. Gelbukh, E. Cambria, Deep learning-based document modeling for personality detection from text, IEEE Intell. Syst. 32 (2) (2017) 74–79.
[28] J. McAuley, J. Leskovec, Hidden factors and hidden topics: understanding rating dimensions with review text, in: Proceedings of the 7th ACM conference on Recommender systems, ACM, 2013, pp. 165–172.
[29] A.Y. Ng, S.J. Russell, et al., Algorithms for inverse reinforcement learning., in: International Conference on Machine Learning, 2000, pp. 663–670.
[30] D. Pfau, O. Vinyals, Connecting generative adversarial networks and actor-critic methods, arXiv preprint arXiv:1610.01945 (2016).
[31] S. Poria, E. Cambria, R. Bajpai, A. Hussain, A review of affective computing: from unimodal analysis to multimodal fusion, Inf. Fusion 37 (2017) 98–125.
[32] G.-J. Qi, Loss-sensitive generative adversarial networks on lipschitz densities, arXiv preprint arXiv:1701.06264 (2017).
[33] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, Int. J. Comput. Vis. 115(3) (2015) 211–252.
[34] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, Improved techniques for training gans, in: Advances in Neural Information Processing Systems, 2016, pp. 2226–2234.
[35] M. Schuster, K.K. Paliwal, Bidirectional recurrent neural networks, IEEE Trans. Signal Process. 45(11) (1997) 2673–2681.
[36] S. Semeniuta, A. Severyn, E. Barth, A hybrid convolutional variational autoencoder for text generation, arXiv preprint arXiv:1702.02390 (2017).
[37] I.V. Serban, A. Sordoni, Y. Bengio, A. Courville, J. Pineau, Building end-to-end dialogue systems using generative hierarchical neural network models, in: Thirtieth AAAI Conference on Artificial Intelligence, 2016, pp. 3776–3784.

[38] I.V. Serban, A. Sordoni, R. Lowe, L. Charlin, J. Pineau, A. Courville, Y. Bengio, A hierarchical latent variable encoder-decoder model for generating dialogues, in: Thirty-First AAAI Conference on Artificial Intelligence, 2017, pp. 3295–3301.
[39] J.T. Springenberg, Unsupervised and semi-supervised learning with categorical generative adversarial networks, arXiv preprint arXiv:1511.06390 (2015).
[40] I. Sutskever, J. Martens, G.E. Hinton, Generating text with recurrent neural networks, in: Proceedings of the 28th International Conference on Machine Learning (ICML-11), 2011, pp. 1017–1024.
[41] R.S. Sutton, A.G. Barto, Reinforcement Learning: an Introduction, 1(1), MIT press Cambridge, 1998.
[42] A. Tjandra, S. Sakti, R. Manurung, M. Adriani, S. Nakamura, Gated recurrent neural tensor network, in: Neural Networks (IJCNN), 2016 International Joint Conference on, IEEE, 2016, pp. 448–455.
[43] D. Vitale, P. Ferragina, U. Scaiella, Classification of short texts by deploying topical annotations, in: European Conference on Information Retrieval, Springer, 2012, pp. 376–387.
[44] T.-H. Wen, D. Vandyke, N. Mrksic, M. Gasic, L.M. Rojas-Barahona, P.-H. Su, S. Ultes, S. Young, A network-based end-to-end trainable task-oriented dialogue system, arXiv preprint arXiv:1604.04562 (2016).
[45] R.J. Williams, D. Zipser, A learning algorithm for continually running fully recurrent neural networks, Neural Comput. 1(2) (1989) 270–280.
[46] Y. Xia, E. Cambria, A. Hussain, H. Zhao, Word polarity disambiguation using bayesian model and opinion-level features, Cognit. Comput. 7 (3) (2015) 369–380.
[47] F. Xing, E. Cambria, R. Welsch, Natural language based financial forecasting: a survey, Artif. Intell. Rev. (2018), doi:10.1007/s10462-017-9588-9.
[48] L. Yu, W. Zhang, J. Wang, Y. Yu, Seqgan: sequence generative adversarial nets with policy gradient, arXiv preprint arXiv:1609.05473 (2016).
[49] Y. Zhang, Z. Gan, L. Carin, Generating text via adversarial training, NIPS workshop on Adversarial Training, 2016.
[50] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, A.A. Efros, Generative visual manipulation on the natural image manifold, in: European Conference on Computer Vision, Springer, 2016, pp. 597–613.