An Improved Approach to Generation and Detection of Out-of-Domain Texts

Bo Wang[1,a)] Tsunenori Mine[1,b)]

Abstract: Many approaches have been proposed to detect Out-of-Domain texts in user intent classification, but most of them are trained on In-Domain data which cannot utilize the huge potential of unlabeled data, or need many hard-to-obtain real Out-of-Domain data. Recently, an Out-of-Domain generation framework has been proposed, which overcame the drawbacks of previous work and got better results. However, the current implementation is far from practical because of its common but ineffective network implementation and unconsidered potential conflicts in the GAN training procedure. In this paper, we propose an improved approach to realize a better Out-of-Domain texts generation, where we modify the Autoencoder for faster learning of context data, and also regularize the output to decrease the difficulty of GAN imitation afterwards. For GAN, we utilize different activation scheme and a more moderate training signal to solve the training conflicts. Comprehensive experiments on three datasets and efficiency measurements show the practicality and efficiency of our new approach.

Keywords: Out-of-Domain detection, text mining, text generation, text classification

1. Introduction

Nowadays, with the development of big data and natural language processing, the deep-learning-based Question and Answering (QA) systems are being widely used. Common QA systems usually use standard question and answer pairs (i.e., In-Domain, IND questions) for network training; however, in real-world applications, unexpected problems (i.e., Out-of-Domain, OOD questions) often arise. Simply ignoring the OOD questions will not only cause misidentifications that degrade the service quality, but can even lead to major problems in some application scenarios. Therefore, it is meaningful and necessary to determine whether a question is from IND or OOD before the intent classification.

Many studies have attempted to build an OOD detector by investigating and adapting an IND classifier. For example, [3] proposed a simple method by using the softmaxed maximum probability of the IND classifier's output as the detection score, while [5] added an entropy term to the loss function and used OOD examples to optimize the OOD-sensitivity of the classifier. Moreover, [18], [20] used the ensemble classifiers to obtain an optimal classification model and got a more proper detection score, and [14], [15] used adversarial training to detect the OOD examples with the Autoencoder and GAN trained on IND data respectively. Recently, as [4] found that the Transformers performed better than classic models, there has been a focus on encoding-and-distance-calculation methods which utilize the large-scale encoder in OOD detection, where [9], [12], [17] reported good results with fine-tuned BERT and its variations as the encoder.

However, in reality, OOD is usually very difficult to obtain, while an adequate environment for training and maintaining a large-scale network is not always affordable for everybody. Meanwhile, there are actually huge unlabeled data like user-input-log generated in daily QA processing, while none of the previous methods can make better use of them.

In these conditions, [19] obtained sentences by using an Autoencoder and a GAN structure, where the encoder maps texts to latent space and the GAN imitates its distribution. [8] also used GAN and IND images as inputs, but added a pre-trained IND classifier as another 'discriminator' to the generator that judges whether the generated image belongs to IND/OOD, which made OOD images generation available. [21] combined these two ideas, added an IND classifier to the text generation framework and used the Shannon entropy loss as the secondary training signal, and finally achieved the generation of fake-OOD texts using IND and unlabeled data.

Despite the success of work [21], after [10]'s reproduce based on the description, the result was far lower than the reported. To solve it, [10] directly abandoned the Autoencoder and used a Sequence-GAN (Seq-GAN), which got a better result. However, we believe that the problems were caused by the primitiveness of [21]'s proposed model. For example, the structure of the Autoencoder is too simple and the training strategy zero-knowledge causes a very slow training process and worsens the decoding effect. The usage of a raw encoding result that forms an uncertain and big encoding latent space burdens the subsequent GAN imitation. In GAN training, there is a conflict between the added strong entropy loss and the original GAN loss that causes unstable procedure.

[1] Kyushu Universtiy, Fukuoka City, Fukuoka 819–0395, Japan
[a)] wangbo.rw@gmail.com
[b)] mine@ait.kyushu-u.ac.jp

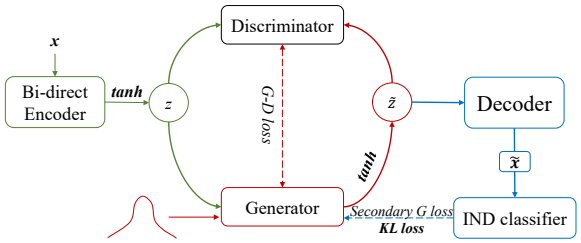


Fig. 1 Our new OOD question generation implementation
Upper: AutoEncoder Bottom: GAN OOD generation

To remedy these issues, for Autoencoder, we deepen the encoder and initialize a pre-trained word embedding to let it gain some prior knowledge. On the other hand, we formalize the output of the Autoencoder and create a smaller latent space to facilitate GAN's imitation process afterward. For GAN, we mainly use a softer loss as the signal for the secondary training of generator to mitigate the conflicts, and also modify the basic structure and activation scheme for a better performance. The overview of our new OOD question generation implementation scheme is shown in Fig. 1.

Finally, by using the data generated with our implementation, the OOD detector is completed. The experimental results demonstrate the superiority of our method.

Our contributions of this study are broadly as follows:

(1) We re-explored and solved the unstable training and poor performance problem of the current adversarial OOD text generation framework. With comprehensive analyses and efficiency-orientated redesigns for each part, we propose a new framework that is truly practical and efficient to be used in real applications.
(2) We provide not only the framework, but a clear and detailed implementation solution from latent space mapping, to OOD question generation, and to the final OOD detector training, which is not available in previous works.
(3) The proposed framework is straightforward and concise, which does not require large-scale networks as the foundation, but can effectively utilize the unlabeled data commonly found in daily application scenarios. With a fast training and detection speed, our model achieves comparable results among state-of-the-arts.

2. The Out-of-Domain generation and detection framework

In this section, we will introduce our proposed framework part by part. First is the OOD question generation part, which consists of two components: an Autoencoder structure to map discrete text sequences into a continuous latent space, and a GAN structure to imitate these textual latent codes. Different from [19], the normal adversarial text generation model, in the GAN training, we use a pre-trained IND question classifier as the second 'discriminator' to supervise the generator: when the generator generates various sentences, the usual discriminator is responsible for real/fake data discrimination, while the IND classifier is expected to 'catch' the grammatically qualified, but more OOD-like generated sentences, and pass the secondary training signal to the generator.

Once the GAN is trained, we obtain fake-OOD questions by decoding the generated latent codes back with the decoder. By utilizing both IND and OOD texts to optimize the classifier from two aspects, we finally complete the OOD detector.

The important assumption that this approach works is that in reality, IND and OOD usually behave similarly in linguistic and grammar, e.g., for an intelligent home assistant, 'what is the hottest TV show at the moment' is an IND question, while 'what is the hottest stock at the moment' is more inclined to an OOD question. We believe these nuances can be captured by our approach. (We will also provide experimental evidence in Sec. 4.2)

2.1 Autoencoder Training

The Autoencoder part consists of an encoder and a decoder. The role of the encoder is to map the discrete text sequences $\{x_1, x_2, ..., x_n\}$ into a continuous latent space $\mathbb{R}^n$ so that the GAN can imitate the distribution afterward, while the decoder reconstructs the words in a sentence given the latent code z.

$$\{x_1, x_2, ..., x_n\} \xrightarrow{encoder} z \in \mathbb{R}^n \xrightarrow{decoder} \{x'_1, x'_2, ..., x'_n, \}$$

The main structure of our Autoencoder can be seen in Fig. 2.

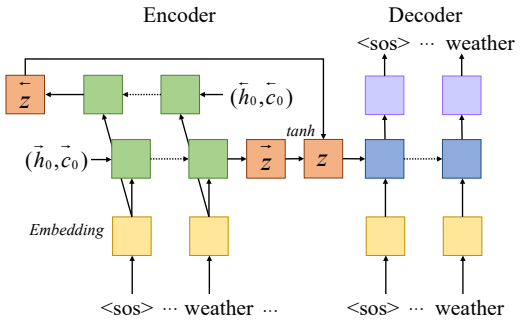


Fig. 2 The structure of autoencoder

In actual implementation, work [19], [21] both used the common single layer, unidirectional RNN (GRU and LSTM [*1] respectively) structure as the Autoencoder. To 'encourage better generalization', during the training procedure, they additionally put an isotropic Gaussian noise $\boldsymbol{I}$ to the latent code z.

However, this noise $\boldsymbol{I}$ is complex as the variance setting is difficult to determine. In our tests, with the effect of the noise, the training procedure of the Autoencoder on such implementations is still not very effective, as the network takes a long time to converge and the decode effect is also not good. For example, we stopped the Autoencoder training with a benchmark of 0.2 or less *reconstruction loss* on both IND and unlabeled data, whereas with the past models, the loss was still 0.27/0.23 after 40 training rounds, while our improved model could reach this standard after only 23 epochs [*2]. In addition, the correct rate of token decoding is only 62.3% on the validation set, which expresses a bad effect on the imitation of GAN.

We located the main reason for the problem as a too simple network structure and the difficulty for the network learning from

[*1] RNN: Recurrent Neural Network, GRU: Gated Recurrent Units, LSTM: Long Short-Term Memory
[*2] measured on CLINC dataset, which we will introduce in Sec. 3.1, the symptoms and comparison in the following sections are also obtained on this dataset.

zero-knowledge. Therefore, we made the following efforts to keep the Autoencoder training stable and efficient, and also reduce some burdens for the following GAN training:

a. Modification on network complexity: since the typical text lengths of questions are usually of medium size, we deepened the former network into a **bi-directional** LSTM. [*3]
b. Giving network a prior knowledge: to make the encoder gain some prior knowledge of the vocabulary and thus easier to train, we used a **pre-trained word embedding** when loading the sequences into the encoder.
c. Regularization & learning difficulty reduction: we added a fully-connected layer and a ***tanh* activation function** at the end of BiLSTM layer. The aim is not only to conduct dimension reduction to fit the decoder's input size, but regularize a smaller latent space to reduce the burden of the decoding and GAN imitation.
d. Others: as the problem was solved with the above methods, we removed the complex but ineffective isotropic Gaussian noise $\boldsymbol{I}$.

In formula, suppose the input sequence is $\boldsymbol{x} = \{x_1, x_2, ..., x_T\}$. At each time step t, the current word x_t first obtains its embedding through the embedding layer of the encoder *ENC*, and then goes through both forward and backward directions of *ENC* along with the previous hidden state (h_{t-1}, c_{t-1}) to get the next state:

$$(h_t, c_t) = ENC(e(x_t), h_{t-1}, c_{t-1}) \tag{1}$$

When the last word x_T was input, the final hidden state (h_T, c_T) is obtained. As the decoder is uni-directional, we used a fully-connected (fc) layer to reduce the dimension to half and a *tanh* function to activate it, which is used as the first state of decoder, and also the latent code z that will be later imitated by GAN:

$$(h'_0, c'_0) = z = tanh(fc((\overrightarrow{h_T}, \overrightarrow{c_T}), (\overleftarrow{h_T}, \overleftarrow{c_T}))) \tag{2}$$

Next, with the initial hidden state (h'_0, c'_0) and the beginning word $x'_0 = \langle sos \rangle$, the decoding procedure starts. The decoder *DEC* decodes out the next word and state, and uses them cyclically until the signal of 'end of the sentence' $x'_T = \langle eos \rangle$ appears.

$$(h'_t, c'_t) = DEC(e'(x'_t), h'_{t-1}, c'_{t-1}) \tag{3}$$

Here, in fact, the decoder does not directly output the next word x'_t, but generates a probability distribution at each time step, where we simply applied the maximum likelihood strategy, i.e., we select the words that have received the highest probability as the result and the next word input:

$$x'_t = max(softmax(fc(h'_{t-1}, c'_{t-1}))) \tag{4}$$

Finally, we train the *ENC* and *DEC* by calculating the reconstruct loss of the original sentence $\boldsymbol{x}$ and decoded sentence $\boldsymbol{x}'$ on both IND and unlabeled data, and update the parameters:

$$\mathcal{L}_{ENC/DEC} = \mathbb{E}_{x \in IND\&unlabel} \left[\mathbb{E}_{i=1 \sim T} - \log(P_{DEC}(x'_i)) \right] \tag{5}$$

[*3] The BiGRU has also been experimented, but the encoding effect is slightly inferior to BiLSTM.

2.2 In-Domain Classifier Training

In order to discriminate whether the generated sentences by the generator are now more inclined to IND or OOD during subsequent GAN training, we need to train an IND classifier in advance.

[21] used a very complex IND classifier (3-MLP plus 4-kernel, 128 feature map for each kernel TextCNN [6] [*4]) to extract the text features more precisely and thus trying to give the generator a more accurate training signal, which put much burden on the full-model operation. As for our solution, because we solved the core problems of the main network, we have vastly streamlined the structure of this part by using a 3-kernel, each kernel 16 feature map TextCNN, so that it can serve as the 'auxiliary' rather than a burden.

This can significantly improve the model efficiency. Taking the floating point operations (FLOPs) computation for one 16-sentence-batch input as an example, the computation needed based on our model (18.39M FLOPs) is **reduced by 84.4%** compared to their model (117.75M FLOPs), not to mention the hundreds of epochs that GAN training typically goes through.

Finally, in formula, given the sentence $\boldsymbol{x}$ and corresponding intent label y from IND dataset, the IND classifier *IDC* is trained with a common cross-entropy loss:

$$\mathcal{L}_{IDC} = \mathbb{E}_{x \in IND} \left[-\boldsymbol{x} \log(P_{IDC}(y = y|\boldsymbol{x}) \right] \tag{6}$$

In the later GAN training, the parameter of *IDC* will be kept fixed for the secondary supervision to the generator.

2.3 Adversarial Training with In-Domain Classifier Discrimination

After the preceding set-ups, we finally come to the core part of the adversarial OOD text generation, where we utilize GAN structure with only IND and unlabeled data to produce fake-OODs.

2.3.1 Main Framework

Firstly, let us shortly introduce the GAN's basic structure. A GAN consists of two components: a generator G and a discriminator D. The generator generates fake latent code $\widetilde{z}$ from random Gaussian noise e, i.e., $\widetilde{z} = G(\varepsilon)$, while the discriminator discriminates the true z (in this study z comes from encoded IND and unlabeled data) from fake $\widetilde{z}$, as shown in Fig. 1 middle part. This competitive learning cycle of G and D constitutes the *adversarial learning process*. When these two networks finally converge, the generator is expected to generate fake $\widetilde{z}$ that is of a similar distribution with z.

The competitive learning and training process of D and G are achieved through Wasserstein distance [2], that is:

$$\mathcal{L}_D = \mathbb{E}_{\substack{\varepsilon \in \mathcal{N}(0,1) \\ z \in IND\&unlabel}} \left[\log D(z) + \log(1 - D(G(\varepsilon)) \right] \tag{7}$$

$$\mathcal{L}_G = \mathbb{E}_{\varepsilon \in \mathcal{N}(0,1)} \left[\log(1 - D(G(\varepsilon)) \right] \tag{8}$$

However, it is obvious that such a GAN can only produce sentences similar to the input, i.e., IND and unlabeled texts, but our generation target is the OOD texts. So following the idea of

[*4] MLP: Multi Layer Perceptron, TextCNN: the Convolutional Neural Network for Text

[8], [21], we added an auxiliary IND classifier (*IDC*) to conduct secondary training to G during GAN training. At this time, the original discriminator is mainly responsible for checking whether the generated latent code is similar to the original distribution, which may be interpreted as a *normal semantic check*. On the other hand, (after the decoder temporarily decodes the latent code back into a sentence, as shown in Fig. 1 right part) the *IDC*, as the second 'discriminator', checks whether the generated text in real-time is more like an IND/OOD text, and passes the classification confidence result as the training signal to the generator, to reinforce it generate more 'boundary' but grammar-certified samples around the IND and unlabeled area in latent space.

Briefly, in formula, in addition to the original training signal from D, we use the output of *IDC* as the secondary training signal for G training:

$$\mathcal{L}'_G = \mathcal{L}_G + \mathcal{L}_{G-IDC}(DEC(\widetilde{z})) \tag{9}$$

2.3.2 Implementation

Up to this point, the idea sounds straightforward and effective, but there are many aspects to consider when it comes to actual implementation. In our pre-experiments, we found that various problems occurred when reproducing existing similar works [8], [21]. The main symptoms are the unstable fluctuation of current $\mathcal{L}'_G$ and $\mathcal{L}_D$, the low confidence of D for G's generated fake data (we could only get $D(G(z))$ of 3%˜4% after training with former implementation), etc., which are far from practical and efficient. For these problems, we conducted the corresponding analysis and solution respectively.

a. Choice of loss as IND classifier training signal $\mathcal{L}_{G-IDC}$:

The reason why the previously described 'IND classifier as a second discriminator' should work is mainly from the phenomenon found in study [3]: when the (generated) input text is more biased towards OOD, the IND classifier will show a lower 'confidence', i.e., the probability output distribution is closer to a uniform distribution.

However, with this property, there comes another problem about the choice of loss that could express this 'lack of confidence'. In work [10], [21], they used a Shannon Entropy loss as the $\mathcal{L}_{G-IDC}$ in eq. (9). However, in pre-experiments, we found that this entropy loss grew significantly as the number of categories increases (sometimes may be several times to original G-D loss $\mathcal{L}_G$), which we believe it leads to a learning objective conflict during training and thus making the training procedure unstable.

After experiments and trials, we finally determined to utilize the KL divergence loss, which does not grow hugely along with the categories, but can more moderately express the distance between current (*IDC*'s output) distribution and target (the uniform) distribution as the training signal $\mathcal{L}_{G-IDC}$, that is,

$$\mathcal{L}'_G = \mathcal{L}_G + \mathbb{E}_{\widetilde{z}\in fake-code}\left[KL(\mathcal{U}\|P_{IDC}(DEC(\widetilde{z})))\right] \tag{10}$$

b. Network structures:

In order to make the GAN learn the features of latent code z as quickly as possible, [19], [21] utilized 5 and 4-MLP networks for D and G respectively, and used *LeakyRelu* to activate each layer by default.

However, in our experiments, we found that merely piling-up the network depth can neither achieve a learning speed-up nor make the network training more stable. Therefore, we simplified the D and G to both 3-MLP, but utilized *Relu* to activate the middle layer of G, which is inspired by DCGAN [13]. On the other hand, as a correspondence to the setting of the Autoencoder (eq. (2)) to lower the imitation difficulty for the generator, we also put *tanh* at the final layer of G. These strategies together guarantee a faster and better convergence process.

c. Auxiliary techniques and other modifications:

In order to offset the above conflicts and instabilities, work [21] used a 'soft-embedding' technique. This method retains a **complete** word decoding probability result of the decoder's output, and calculates the average embedding as the embedded input to the decoder, which brings a huge amount of computation and is also space consuming. They also used 'temperature scaling' to 'sharpen' the output of the decoder. But for a similar reason that as the core problems have been largely solved through our proposed improvements, we removed these two redundant auxiliary methods. Instead, as mentioned in Sec. 2.1, we continued to use maximum-likelihood to reconstruct the words in a sentence, which also improves the efficiency.

Other performance and stability issue includes the fact that [19], [21] enabled a complex gradient penalty item for GAN as default, which is reported in [2] to be effective in general GAN training. But in our experiments, we found that this term also has a tendency to conflict with current $\mathcal{L}_{G-IDC}$, so we removed it.

2.4 Out-of-Domain Detector Training

After the GAN finishes training, we are able to generate a large number of fake OOD latent codes. By decoding them with the decoder, the fake OOD sentences can be obtained:

$$\widetilde{\boldsymbol{x}} = DEC(\widetilde{z}) = DEC(G(\varepsilon))$$

In order to train the Out-of-Domain Classifier (*ODC*), we used a similar strategy as [5], i.e., we used both the IND and fake-OOD texts to optimize the *ODC* in two aspects: the IND data are used to ensure a basic and necessary prior semantic knowledge, while the OOD data focus on reinforcing the detector to give a more uniformed output when encountering suspected OOD examples, specifically:

$$\begin{aligned}\mathcal{L}_{ODC} &= \mathcal{L}_{ODC-IND} + \mathcal{L}_{ODC-fakeOOD} \\ &= \mathbb{E}_{\widetilde{\boldsymbol{x}}\in IND}\left[-x_i \log\left(P_{ODC}(y = y_i|\boldsymbol{x})\right]\right. \\ &\quad \left. + \mathbb{E}_{\widetilde{\boldsymbol{x}}\in fake-OOD}\left[KL(\mathcal{U}\|P_{ODC}(\widetilde{\boldsymbol{x}}))\right)\right]\end{aligned} \tag{11}$$

Note that in work [5], the authors utilized the entropy loss as the $\mathcal{L}_{ODC-fakeOOD}$, the input data for this part are also real OOD texts, while we continued to utilize a softer KL loss and use only our own network-generated fake OOD examples to train *ODC*.

Finally, following [3]'s approach, we passed the probabilistic

output of *ODC* through a softmax layer, and took the maximum value as the final detection score. It is expected that the reinforced OOD detector will output ***a lower score*** when encountering real OOD sentences.

3. Experiments

To demonstrate the generality and efficiency of our model, we select three public datasets and a series of highly competitive methods as the baselines to conduct experiments.

3.1 Datasets

Here are the datasets used in our experiments. Since we found only one dataset called CLINC specifically designed for OOD detection, we additionally adapted the other two IND user-intention classification datasets into OOD detection. In addition, because there exists no dataset providing the corresponding unlabeled data, ***we mixed a large amount of IND data and a small portion of OOD data to simulate unlabeled data***. The composition and partition information is shown in Table 1.

Table 1 Data composition and partition details

	CLINC	SNIPS	FB-Multi
Vocab. size	7,259	10,335	2,517
IND intent num.	150	6	10
IND train rec.	18,000	11,237	12,003
OOD intent	original	'play music'	'show reminder'
OOD train rec.	350	250	250
Mixdata comp.	*ind*9,900+*ood*350	*ind*6,000+*ood*250	*ind*6,000+*ood*250
Test data rec.	5,500	1,400	4,395
Test data comp.	*ind* 4,500 *ood* 1,000	*ind* 1,214 *ood* 186	*ind* 3,829 *ood* 566

CLINC dataset[7]: it has 22,500 IND data in 150 intents and 1,200 OOD data, balanced in each category. They also provide a complementary dataset with 150 additional OOD training data — OOD+, which we use this time.

SNIPS dataset[1]: collected from the Snips voice platform, which contains a balanced 15,000 texts in 7 intents. We manually choose 'play music' as the OOD category because of its similar but different nature to another 'add to playlist' category. We believe this will make the OOD detection task even more difficult and can better reflect the superiority of our model.

FB-Multi dataset[16]: allocated from the Facebook multilingual database which we only use its English part. It contains over 35,000 data in 12 intents and the distribution is unbalanced. Because the quantity of 'weather find' category is huge (over 10K records), which not only causes data imbalance, but contains many noises as the place names around the world, we remove this category. As there exist very similar categories 'set reminder' and 'cancel reminder', we select 'show reminders' as the OOD category.

3.2 Baselines

We apply these competitive methods as our baselines, where *a)˜d)* belong to the classifier-output-based methods, *e), f)* belong to the encoding-and-distance-computing methods, *g), h)* represent the other two adversarial-based methods, and *i), j)* are two simple methods for generating fake-OODs.

a) IDC-MSP[3]: the output of a pre-trained IND classifier (*IDC*) is activated by $softmax$ function, and the maximum value of the output probability is used as the detection score.

b) BERT-MSP[12]: a similar method to *IDC-MSP*, but the IND classifier is utilized with a BERT network, which is fine-tuned with IND data in advance.

c) IDC-Entropy: the output of *IDC* is activated by $softmax$ function, and the *negative* value of the entropy on this output is used as the detection score.

d) IDC-KL Dist.: a similar method to *IDC-Entropy*, but the KL divergence between the output and the uniform distribution is calculated as the detection score.

e) BERT-Euclid: firstly a BERT is fine-tuned with IND training set, then the vectors on [CLS] position (i.e. the sentence vector) of all texts from the BERT's output are extracted. The *negative* value of shortest Euclidean distance between test texts and IND training texts is used as the detection score.

f) BERT-Maha[12]: a similar method to *BERT-Euclid*, but the Mahalanobis distance is calculated.

g) GAN-DIS [15]: a GAN structure is trained only on IND data and the discriminator is extracted. The output value of the discriminator on the test set is used as the detection score.

h) En/De Recons. [14]: an AutoEncoder is trained only on IND data. After training, the *negative* value of reconstruction loss on the test set is used as the detection score.

i) Del N as fOOD: a simple OOD oversampling method that removes all the nouns from training data and treats the rest as fake OOD data.

j) Del V as fOOD: all the verbs in the sentences are removed this time and the rest is used as fake OOD data.

In actual implementation, we used the same TextCNN as described in Sec. 2.2 as the base IND classifier for methods *a), c), d)*. To represent the on-trend BERT-based methods while considering the reproduce difficulty, we used the simplest pre-trained BERT network *'bert-base-uncased'* [*5] as the base network to conduct *b), e), f)*, where we would like to show (in Sec 4.3) that although some of the strengthened BERT-based models may exceed our results in some metrics, we have a substantial advantage in terms of efficiency over even the simplest implementation. We use the same Autoencoder and GAN structure described in Sec. 2.1 and 2.3 as the base networks for methods *g), h)*, respectively.

In addition, as the two similar adversarial OOD text generation works [10], [21] also used the CLINC dataset, we add them to our baselines as well.

3.3 Metrics

Most OOD detection methods follow a detection score (e.g. in Fig. 4) - threshold judgment approach, that is,

$$this\ question\ is\ an \begin{cases} IND,\ if\ d_{score} \geq thres. \\ OOD,\ if\ d_{score} < thres. \end{cases}$$

However, once the threshold changes, the results/accuracy will also vary. Coupled with the imbalanced quantity fact of

[*5] https://huggingface.co/transformers/pretrained_models.html

IND/OOD data in the test set, we use four metrics that are neither affected by the threshold nor the data balance.

AUROC and AUPR: i.e., *Area Under the Receiver Operating Characteristic curve* and *Area Under Precision-Recall curve*. For AUROC, the area enclosed by the false positive rate (*FPR*)- true positive rate (*TPR*) curve under different threshold is calculated, while for AUPR it is the area enclosed by the Precision-Recall curve. Both of them are able to effectively measure the model performance even when the test data are not balanced. Both are the higher the better.

FPR95 and FPR 90: to show the practical performance of the model, we also apply FPR95 and FPR90, which denote the FPR values when the TPR reaches 95% and 90%, respectively. Both are the smaller the better.

3.4 Implementation

Here we introduce our detailed implementation method following the description order in Sec. 2. First, for the Autoencoder part, we used a BiLSTM-LSTM structure, both of them contain one hidden layer of 300 units. Before training, we used a fasttext [11] English 300-dimensional pre-trained word vector to initialize the embedding layer of the encoder. In addition, at the end of BiLSTM layer, we used a fully connected layer to reduce the dimension by half for the final hidden state and activated it with *tanh*.

For the IND classifier, we used a [3,4,5] kernel size, 16 feature maps for each kernel TextCNN. Note that the final OOD classifier (*ODC*) also shares this structure, but is optimized from both sides with IND and fake-OOD input. Finally for GAN, we used two 3-MLPs as the generator and discriminator, the dimensions of each layer are 600. For *G*, we used *Relu* function to activate the middle layer, and also added *tanh* in the final layer corresponding to the encoder setting. For *D*, we used *LeakyRelu* activation function throughout the layers.

4. Results and discussion

In this section, we demonstrate the effectiveness of our proposed method using various datasets. Due to the space limitation, we will mainly make discussions on the CLINC dataset after reporting all experiment results, for it is the most commonly used dataset in OOD detection works.

4.1 Results and Analysis

The results of each method for OOD detection on the three datasets can be seen in Tables 2,3,4 respectively, where our approach achieved the best results on all datasets among the other baselines. Although work [21] reported a good result, after our (and also [10]'s) best effort to reproduce, it only achieved an AUROC of 88.87. On the other hand, it can be seen that with our new implementation, this framework shows its real strength. At the same time, some simple methods such as directly calculating the KL distance have also obtained good results. Meanwhile, the method *BERT-Euclid* showed great strength and achieved almost tie results with us on *CLINC* and *FB-Multi* datasets. However, this baseline took a long time calculating distances, whose efficiency may be lower in real applications (we will discuss this again in Sec. 4.3). In addition, on the *FB-Multi* dataset, the *BERT-Maha* obtained much worse results compared to *BERT-Euclid*. We consider the main reason as the unbalanced data composition of *FB-Multi*, which may cause a higher error in calculating the covariance matrix.

Table 2 Results on CLINC dataset in %

Method	AUROC↑	AUPR↑	FPR95↓	FPR90↓
IDC-MSP	92.83	98.30	38.3	20.6
BERT-MSP	93.73	98.18	22.9	14.7
IDC-Entropy	93.98	98.55	27.1	16.2
IDC-KL Dist.	94.12	98.54	24.9	13.4
BERT-Euclid	95.70	98.89	17.4	10.0
BERT-Maha	94.41	98.56	24.0	13.4
GAN-DIS.	53.04	84.11	94.6	89.5
En/De Recons.	83.40	95.97	71.5	54.8
Del N as fOOD	91.86	97.82	30.1	21.1
Del V as fOOD	93.77	98.38	26.0	14.8
Old impl. reported[21]	95.83	99.05	23.7	9.5
Our reproduce of [21]	88.87	97.29	53.9	30.6
[10]'s reproduce of [21]	88.79	58.22	36.5	26.9
Seq-GAN [10]	91.24	97.79	26.1	19.3
ODC-MSP(ours)	**96.17**	**99.09**	**16.8**	**8.9**

Table 3 Results on SNIPS dataset in %.

Method	AUROC↑	AUPR↑	FPR95↓	FPR90↓
IDC-MSP	91.96	98.78	47.8	22.6
BERT-MSP	92.52	98.85	47.3	24.2
IDC-Entropy	92.10	98.79	45.7	22.6
IDC-KL Dist.	94.09	99.10	32.3	14.5
BERT-Euclid	96.63	99.47	11.3	3.8
BERT-Maha	94.48	99.17	26.9	8.1
GAN-DIS.	68.31	92.64	81.7	73.1
En/De Recons.	69.69	94.00	84.4	73.7
Del N as fOOD	94.78	99.20	25.3	14.0
Del V as fOOD	74.78	95.77	91.4	80.1
ODC-MSP(ours)	**98.74**	**99.83**	**4.3**	**1.8**

Table 4 Results on FB-Multi dataset in %.

Method	AUROC↑	AUPR↑	FPR95↓	FPR90↓
IDC-MSP	93.69	99.01	29.2	11.5
BERT-MSP	84.97	97.65	77.4	53.0
IDC-Entropy	93.87	99.03	30.2	11.0
IDC-KL Dist.	94.71	99.18	30.0	14.5
BERT-Euclid	98.07	99.69	24.7	19.4
BERT-Maha	89.41	98.42	66.1	42.8
GAN-DIS.	70.88	97.47	88.0	72.3
En/De Recons.	59.97	96.79	94.0	89.2
Del N as fOOD	90.66	98.34	40.1	22.4
Del V as fOOD	92.70	98.85	36.9	15.7
ODC-MSP(ours)	**98.15**	**99.71**	**5.8**	**3.2**

Fig. 3 shows the ROC curve patterns for different methods (whose enclosed area is the AUROC), which can confirm again that our model outperforms all baselines. Although the *IDC-MSP* method had a strong start, it performed worse later. In addition, *BERT-Euclid*, a strong competitor, performed slightly worse than our model in the beginning but caught up afterward.

Next, Fig. 4 shows the distribution of detection scores for the baselines, where we can clearly observe the different detection styles of each method (Note that *a lower score* indicates the classifier considered the example more tends to be OOD): The *IDC-MSP* had strong confidences in classifying IND data and output high detection scores, but it gave relatively average scores to OOD data, while *BERT-Eucild* tried to keep giving higher scores

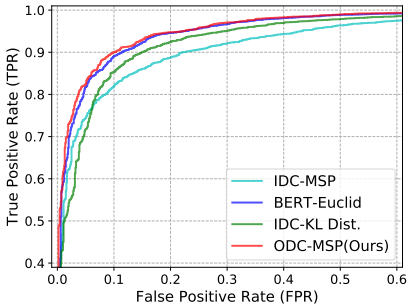
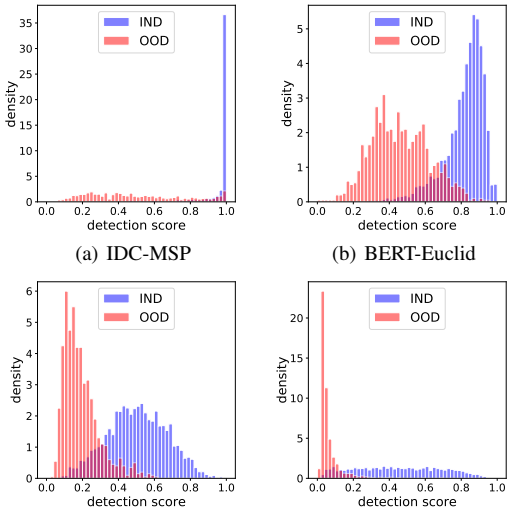


Fig. 3 ROC curves on CLINC dataset

for IND data and scored lower for OOD, which is a good pattern. The *IDC-KL Dist.* method, which directly utilized the KL divergence as scores, exhibited a nearly opposite pattern to *BERT-Euclid*, i.e., more focusing on detecting and giving low scores to OOD data. Finally, as for our model, although probably no longer suitable to serve as an IND classifier, exhibited a strong sensitivity to OOD examples and output very low detection scores for almost all OOD data. We believe this is what makes our model outperformed the others.



(a) IDC-MSP (b) BERT-Euclid

(c) IDC-KL Dist. (d) ODC-MSP(Ours)

Fig. 4 Detection score distribution on CLINC dataset

4.2 Case Study and Generated Data Analysis

Here are some examples of the text generated by our adversarial framework on the CLINC dataset, as Table 5 shows.

Table 5 Generated examples on CLINC dataset

Good OOD-like sentences	teach me how to book the chicken in happiness where the i lost my mastercard use the gas amount to get my gas bill
IND-like sentences	please check and see why my card was declined at amazoncom
Noise (meaningless)	what time is the wait to change pizza
Real IND	can you help me understand why my card got declined what time will i get to the beach taking the bus
Real OOD	tell me the steps as to how to begin a career as a journalist where can i find the best divorce lawyer

We could notice that not all generated texts are perfect fake-OODs, some are more like IND texts, while some others may look like noise. Although only a subset of good fake-OOD texts was generated, we convince it is because this subset, which successfully captured a similar pattern with real OOD questions, contributes to the better result. We believe *the quality of generated text will be even better as we apply more real unlabeled data* in the future.

To quantitatively indicate the quality of the generated texts, we put 18K IND, 350 real OOD texts from the training dataset and generated 54K fake-OOD texts into the IND classifier trained in Sec. 2.2 respectively, and calculated their KL divergence to the uniform distribution. The result is shown in Fig. 5.

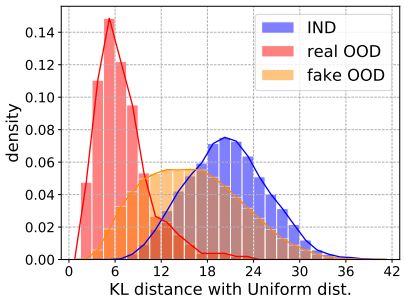


Fig. 5 KL distance analysis on CLINC dataset

Remember the phenomenon reported by [3] that, if an example is closer to OOD, the output of IND classifier should be closer to a uniform distribution, i.e., the KL distance should be smaller. As it happens, we can observe from the figure that the fake-OOD examples generated by our model, though which may still be not as good as the real OOD, have had a significantly smaller KL distance than the IND examples, which also demonstrates that the sentences generated with our method are transforming towards a more OOD-like direction.

4.3 Efficiency Comparison

Finally, we compare the efficiency of the models with the parameters and time costed during training and predicting on CLINC dataset. The results are summarized in Table 6.

Table 6 Efficiency comparison on parameter and time

Method	Total Para.	Pred Para.	Train Time	Pred Time	AUROC
IDC-Based	2.24M		18s	avg 0.90s	best 94.12
BERT-MSP	110M		2min20s	2.48s	93.73
BERT-Euclid				12min	95.70
BERT-Maha				45min	94.41
GAN-DIS	1.80M	0.72M	1min43s	0.28s	53.04
En/De Recons.	9.06M		3min38s	0.94s	83.40
Work [21]	16.06M	5.63M	1h31min	1.38s	88.87
Our model	15.34M	**2.24M**	11min27s	0.93s	**96.17**

Note: the train time of BERT denotes the costed time of 4-epoch finetuning, the pred time includes both encoding and distance calculation steps.

The efficiency advantage of our model can be clearly seen in this table. Although the *GAN-DIS* model had the least parameters and overall time, it also obtained the worst result. The finetuning time of *BERT-Euclid* was relatively short, also it only took 5.06s to encode 5,500 test sentences, the results are also very competitive; however, it spent huge time on the 768-dimension distance calculation and comparison with all the data in the database when making predictions, which is not very practical in real applications. The original implementation of [21] spent much time on training, especially on the GAN training with over 1h16min

costed, where we consider the reason as the usage of big auxiliary *IDC* network and 'soft-embedding' along with huge extra computations. Our method, on the other hand, although required slightly more parameters and training time compared to classical methods, improved significantly in terms of final result and prediction speed. In conclusion, we believe our model is well-balanced and maintains the best efficiency.

5. Conclusion and future work

In this paper, we re-explored and solved the unstable training and poor performance problem of the current adversarial OOD text generation framework, and proposed a new implementation scheme. For Autoencoder, we deepened the encoder and added a *tanh* for a normalized latent space to release some difficulties of GAN imitation. For the unstable training problem of GAN, as a replacement of conflicting entropy loss, we used a softer KL divergence loss to transmit the signal from IND classifier to the generator, and we refined the basic structure for a quicker grasp of the input distribution. The results of experiments and comparisons demonstrated the practicality and efficiency of our model.

As future work, we present a challenge and a refinement direction: the challenge is that due to the natural limitations of the GAN, if IND and OOD are very different, our OOD generation model may lose its advantage. In addition, as for the most important part of the generation framework, the use of new, large-scale-network encoders has become a trend. But as we analyzed, this will greatly reduce the efficiency. We consider using the latest, simplified networks as a pure-encoder in the future to get better coding space distributions but keep a good efficiency. Finally, we would also like to evaluate our model utilizing the real unlabeled data when it becomes available in the future.

Acknowledgment

This research was partially supported by JSPS KAKENHI No. JP20H04300 and JP21H00907. We also thank Seita Shimada and Yoshiaki Matsukawa, Rakuten Card Co. ltd. for their helpful comments on this research.

References

[1] Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. Snips Voice Platform: An embedded Spoken Language Understanding system for private-by-design voice interfaces. *arXiv*, 2018.

[2] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5769–5779, 2017.

[3] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017.

[4] Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, and Dawn Song. Pretrained transformers improve out-of-distribution robustness. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2744–2751, Online, July 2020. Association for Computational Linguistics.

[5] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. In *International Conference on Learning Representations*, 2018.

[6] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics.

[7] Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1311–1316, Hong Kong, China, November 2019. Association for Computational Linguistics.

[8] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *International Conference on Learning Representations*, 2018.

[9] Jeremiah Zhe Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax-Weiss, and Balaji Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *arXiv*, 2020.

[10] Petr Marek, Vishal Ishwar Naik, Vincent Auvray, and Anuj Goyal. OodGAN: Generative Adversarial Network for Out-of-Domain Data Generation. *arXiv preprint arXiv:2104.02484*, 2021.

[11] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA).

[12] Alexander Podolskiy, Dmitry Lipin, Andrey Bout, Ekaterina Artemova, and Irina Piontkovskaya. Revisiting Mahalanobis Distance for Transformer-Based Out-of-Domain Detection. *arXiv preprint arXiv:2101.03778*, 2021.

[13] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016.

[14] Seonghan Ryu, Seokhwan Kim, Junhwi Choi, Hwanjo Yu, and Gary Geunbae Lee. Neural sentence embedding using only in-domain sentences for out-of-domain sentence detection in dialog systems. *Pattern Recognition Letters*, 88:26–32, 2017.

[15] Seonghan Ryu, Sangjun Koo, Hwanjo Yu, and Gary Geunbae Lee. Out-of-domain detection based on generative adversarial network. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 714–718, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[16] Sebastian Schuster, Sonal Gupta, Rushin Shah, and Mike Lewis. Cross-lingual transfer learning for multilingual task oriented dialog. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3795–3805, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[17] Yilin Shen, Yen-Chang Hsu, Avik Ray, and Hongxia Jin. Enhancing the generalization for intent classification and out-of-domain detection in slu. *arXiv preprint arXiv:2106.14464*, 2021.

[18] Lei Shu, Hu Xu, and Bing Liu. DOC: Deep open classification of text documents. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2911–2916, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[19] Sandeep Subramanian, Sai Rajeswar, Alessandro Sordoni, Adam Trischler, Aaron Courville, and Christopher Pal. Towards text generation with adversarially learned neural outlines. In *Advances in Neural Information Processing Systems*, volume 2018-December, pages 7551–7563, 2018.

[20] Apoorv Vyas, Nataraj Jammalamadaka, Xia Zhu, Dipankar Das, Bharat Kaul, and Theodore L Willke. Out-of-distribution detection using an ensemble of self supervised leave-out classifiers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 550–564, 2018.

[21] Yinhe Zheng, Guanyi Chen, and Minlie Huang. Out-of-domain detection for natural language understanding in dialog systems. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:1198–1209, 2020.