OodGAN: Generative Adversarial Network for Out-of-Domain Data Generation

Petr Marek*
Czech Technical University in Prague
Prague, Czech Republic
marekp17@fel.cvut.cz

Vishal Ishwar Naik
Amazon Alexa AI
Sunnyvale, California
naikvish@amazon.com

Vincent Auvray
Amazon Alexa AI
Sunnyvale, California
vauvray@amazon.de

Anuj Goyal
Amazon Alexa AI
Sunnyvale, California
anujgoya@amazon.com

Abstract

Detecting an Out-of-Domain (OOD) utterance is crucial for a robust dialog system. Most dialog systems are trained on a pool of annotated OOD data to achieve this goal. However, collecting the annotated OOD data for a given domain is an expensive process. To mitigate this issue, previous works have proposed generative adversarial networks (GAN) based models to generate OOD data for a given domain automatically. However, these proposed models do not work directly with the text. They work with the text's latent space instead, enforcing these models to include components responsible for encoding text into latent space and decoding it back, such as auto-encoder. These components increase the model complexity, making it difficult to train.

We propose OodGAN, a sequential generative adversarial network (SeqGAN) based model for OOD data generation. Our proposed model works directly on the text and hence eliminates the need to include an auto-encoder. OOD data generated using OodGAN model outperforms state-of-the-art in OOD detection metrics for ROSTD (67% relative improvement in FPR 0.95) and OSQ datasets (28% relative improvement in FPR 0.95) (Zheng et al., 2020).

1 Introduction

OOD detection is an essential task in AI voice assistants like Alexa, Siri, or Google Assistant. The task is to recognize whether a given user utterance belongs to the in-domain (IND) distribution or not. Users usually do not know the limitations of a voice application and assign requests which the system can not act upon. These requests are referred to as OOD since these do not belong to the application's domain. Voice assistants should be able to handle OOD utterances robustly by not taking unintended action or giving wrong or nonsensical responses leading to a poor user experience.

Intent classification (IC) is one of the main tasks in a conversational system that selects the best intent given a user input. IC can be extended to support OOD detection in two different ways. The first one is to add OOD as another intent to the IC model, but this requires annotated OOD data for training. The second method is to use a threshold on the classifier's output probability distribution during the runtime. This method does not require OOD data for training necessarily. Nevertheless, it proves difficult to select the threshold in practice without it.

The state-of-the-art IC algorithms are trained using neural networks to produce probability distribution over output classes and use cross-entropy loss. However, Lakshminarayanan et al. (2017), and Guo et al. (2017) pointed out that the neural network classifier tends to be overconfident in its classification. This means that the classifier tends to assign a high probability for one class, even when the example was not seen in the training phase. Thus, such a classifier cannot correctly recognize if an example belongs to an IND or OOD distribution during runtime with any reasonable threshold value. In this paper, we focus on improving the performance of the threshold-based OOD detection method with the help of generated OOD data.

Zheng et al. (2020) proposed to use negative entropy as an additional loss for the classification task in a neural network. The negative entropy loss trains the network to flatten the produced probability distribution as opposed to cross-entropy, which teaches the network to maximize the correct class probability. Thus, the idea is to apply cross-entropy loss on IND data and negative entropy loss on OOD data. The result is that IND data receives a high

* Research conducted during an internship at Amazon Alexa AI

probability for the correct class, and OOD data receives low probabilities for all classes. Thanks to this fact, we can select a reasonable threshold on the output probability that will classify both IND and OOD data correctly. We need OOD data to train models in this way. However, the collection of OOD data is a manual and expensive process.

The IND data forms a small distribution cluster in the space of vector text representation. In principle, the rest of that space is covered by OOD data. Also, in real-world scenarios, most OOD data share patterns with IND data. Nevertheless, Zheng et al. (2020) demonstrated that training IC model with OOD data that are just outside IND distribution should be sufficient to handle most of the OOD requests during runtime.

In this paper, we propose a novel OOD data generation model OodGAN, which is an extension of SeqGAN (Yu et al., 2017). We use GAN to generate OOD data that share the same patterns as IND and are very close to IND distribution.

Our proposed model aims to be deployed to Natural Language Understanding (NLU) frameworks offered by popular voice assistants like Amazon Alexa and Google Assistant. These NLU frameworks are offered to third-party developers to create voice applications. Third-party developers can define any number of IND intents and provide sample utterances for each to build voice applications. These voice applications should recognize OOD requests during run time without additional developer effort to provide OOD training data. The proposed model can be deployed in a NLU framework to generate application-specific OOD data that the IC model can use during training to recognize OOD requests robustly and improve the end-user experience.

Our main contributions are:

(1) We propose a novel and simple OOD data generation model OodGAN that improves on the model proposed by Zheng et al. (2020). It works with a sequence of words directly unlike the previously proposed models, which work on latent space represented by auto-encoder. Our model eliminates the need for the auto-encoder, which reduces the overall size of the model.

(2) We evaluate our model on the ROSTD and OSQ datasets, and we show that OOD examples generated by OodGAN achieved state-of-the-art results.

2 Related Work

There are three research areas relevant to our work: OOD detection, text generation and OOD generation.

Out-of-Domain Detection

Larson et al. (2019) introduced a dataset for intent classification that includes OOD queries. They propose three baseline approaches for OOD detection that rely on OOD training data. Gangal et al. (2019) created a ROSTD dataset and explored likelihood ratio based approaches. Lee and Shalyminov (2019) proposed an OOD detection method that does not require OOD data by utilizing counterfeit OOD turns in the context of a dialog. Ryu et al. (2018) proposed an OOD detection system that uses only IND sentences to build a generative adversarial network in which the discriminator generates low scores for OOD sentences.

Text Generation

Donahue and Rumshisky (2018) proposed a two-step solution to text generation using auto-encoder and GAN that works with a low-dimensional representation of sentences. Yu et al. (2017) proposed a sequence generation framework SeqGAN that works directly on the text and hence eliminates the need for an auto-encoder.

Out-of-Domain Data Generation

Zheng et al. (2020) proposed a GAN based model to generate pseudo-OOD examples that are akin to IND input utterances. The model uses a denoising auto-encoder that is trained to map an input example into a latent code. The functions of the auto-encoder's parts are the following. The encoder learns to create a latent representation of the examples. The decoder learns to convert the vector of the latent representation into text. The model's generator produces vectors in the latent space. The discriminator evaluates the closeness of latent space vectors generated by the generator to real latent space vectors created by the encoder. Discriminator sends a training signal to the generator to force it to generate indistinguishable vectors from vectors encoded by the encoder. An auxiliary classifier trained on IND examples is introduced to force the generator to generate latent code belonging to OOD. The resulting utterances share patterns with IND examples but belong to OOD.

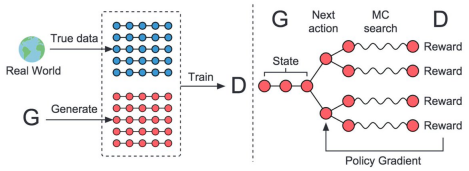


Figure 1: The illustration of SeqGAN (Yu et al., 2017). Left: Discriminator D is trained over the real data and the data generated by generator G. Right: Generator is trained by policy gradient where the final reward signal is provided by the discriminator and is passed back to the intermediate action value via Monte Carlo search.

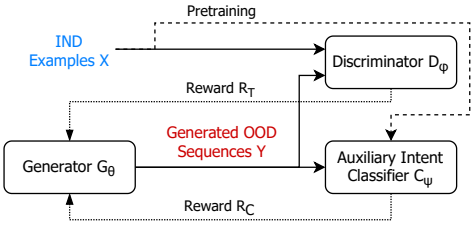


Figure 2: The overall architecture of the OodGAN. C_ψ is pretrained to recognize intent classes for IND examples. D_ϕ is trained to distinguish between IND and generated OOD examples during adversarial training. G_θ is trained by the REINFORCE algorithm during adversarial training to generate OOD sequences. The training is guided by rewards originating in C_ψ and D_ϕ.

3 Generative Adversarial Networks for Out-of-Domain Data Generation

3.1 SeqGAN

The SeqGAN model proposed by Yu et al. (2017) is a starting point for the proposed OodGAN. SeqGAN is a sequence generation framework illustrated in Figure 1. Yu et al. (2017) denote the problem of sequence generation as follows. Given a dataset of real-world structured sequences, train a θ-parameterized generative model G_θ to produce a sequence $Y_{1:T} = (y_1, ..., y_t, ...y_T), y_t \in Y$ where Y is the vocabulary of candidate tokens. They apply reinforcement learning to this problem. In timestep t, the state s is the current produced tokens $(y_1, ..., y_{t-1})$ and the action a is the next token y_t to select.

They propose to additionally train a ϕ-parameterized discriminative model D_ϕ that provides guidance for improving generator G_θ. D_ϕ produces a probability $D_\phi(Y_{1:T})$ representing the probability of $Y_{1:T}$ being a real sequence vs. a generated one. The discriminative model D_ϕ is trained with real sequence data, labeled as positive examples, and synthetic sequences from the generative model G_θ, labeled as negative examples.

SeqGAN uses the REINFORCE algorithm (Williams, 1992) to train generative model G_θ. Parameters of generative model G_θ are updated at the same time by a policy gradient and Monte Carlo search based on the expected end reward received from the discriminative model D_ϕ for the generated sequence. The reward is represented by a likelihood that the generated sequence will fool the discriminative model D_ϕ. Thus the generator's goal is to generate a sequence that would fool the discriminator into considering it as real.

3.2 OodGAN

We propose OodGAN based on SeqGAN. There are two benefits of SeqGAN for our task of OOD data generation. SeqGAN produces sequences similar to the training data, and it works directly on input sequence unlike earlier model (Zheng et al., 2020), which works on latent space. Eliminating the auto-encoder responsible for converting a sequence of words into latent space reduces the overall model size. Also, our experiments with Zheng et al. (2020) show a degradation in the overall performance due to the auto-encoder component (see the Results section for details).

Since our task is to generate OOD data, we have the additional criterion that generated sequences should be close to the training IND sequences. However, we also want them not to belong to any IND intent class. We propose the OodGAN to achieve the two criteria.

The main difference between SeqGAN and OodGAN is the introduction of an auxiliary intent classifier. The auxiliary intent classifier C_ψ estimates the probability $C_\psi(z_i|Y)$ of example Y belonging into intent class z_i. The task of the auxiliary intent classifier is to produce an additional reward signal. The reward signal guides the generator to produce a sequence not belonging to any IND intent class. The reward R_{C_ψ} coming from the auxiliary intent classifier for each generated example is defined as Shannon's Entropy $R_{C_\psi} = -\sum_{i=1}^{m} C_\psi(z_i|Y) \cdot log(C_\psi(z_i|Y))$, where m is the number of IND intent classes. The intuition for using Shannon's Entropy is that we want to reward a generator for producing examples for which the auxiliary intent classifier cannot clearly assign one of IND classes. In other words, the auxiliary classifier should assign a nearly uniform distribution across all intent classes for a good generated

example. The generator obtains a high reward for such examples because the uniform distribution has the highest Shannon's Entropy.

We train the auxiliary intent classifier to predict one of the classes $z_{1...m}$ for each training IND example $X_{1...n}$ during the pre-training step. We do not have to retrain it during adversarial training because IND intent classes' distribution does not change.

The goal of the generator is to generate a sequence that maximizes the expected sum of rewards from discriminator D_ϕ (the estimated probability of the sequence being real), and auxiliary intent classifier C_ψ (Shannon's Entropy calculated using estimated probabilities of sequence belonging to IND intent classes by auxiliary intent classifier).

Empirically, we evaluated different training strategies. We found that optimizing generator G using only the discriminator's reward first, followed by using only the auxiliary intent classifier reward, and then repeating the process for each training batch produced the most stable results. This worked better than summing up the rewards from the discriminator and auxiliary intent classifier. When we tried summing up the two rewards, we noticed that the generator tended to collapse into a state in which it generated a single sequence highly rewarded by the auxiliary intent classifier, even though this did not happen for all training runs. We observed this situation even when we normalized rewards to a value between 0 and 1.

We also observed that part of the examples generated by OodGAN is semantically similar to some IND training example or is generated multiple times. Examples that are identical or too close to IND examples are problematic and confuse the OOD classifier. Duplicated examples do not represent the OOD distribution effectively. For those reasons, we removed with an automatic filter the generated OOD examples that are identical or similar to IND examples or that are generated repeatedly.

To summarize, OodGAN's training procedure has the following steps.

(1) Train Auxiliary classifier: First train auxiliary classifier to predict the classes $z_{1...m}$ for IND data $X_{1...n}$ until convergence.

(2) Train Generator as Language Model: Next, train the generator on the IND data $X_{1...n}$ as a language model until it converges. Thanks to this step, it is easier for the generator to fool the discriminator from the start of the adversarial training.

(3) Train Discriminator: Generate adversarial examples from the generator. This training step helps the discriminator to provide a useful reward signal from the start of adversarial training.

(4) Adversarial Training: Perform adversarial training of generator and discriminator. There are three optimization steps for each training batch. First, optimize the generator using reward from discriminator as proposed by Yu et al. (2017). Next, optimize the generator using a reward from the auxiliary classifier. Lastly, optimize the discriminator.

4 Experiments

4.1 Datasets

We conducted experiments on ROSTD (Gangal et al., 2019) and OSQ (Larson et al., 2019) datasets.

- **ROSTD** contains three categories (alarm, reminder, and weather), each consisting of four intents. The dataset consists of 30,000 training, 4,000 validation and 8,000 testing IND examples. OOD examples were selected in a way that they do not belong to any category and do not share patterns with any IND examples. There are also no OOD examples in the training set of the dataset. The testing set contains 4,500 OOD examples. IND and OOD examples from ROSTD are listed in Table 5.

- **OSQ** consists of 150 intents. The datases consists of 15,000 training, 3,000 validation and 4,500 testing IND examples. The dataset was created using Mechanical Turk. The turkers were given the name of the intent, and they were supposed to write intent examples fitting into the intent. The dataset authors manually went through examples and moved examples not fitting into the given intent class to the OOD class. In this way, OOD examples share the same patterns as IND examples. The OSQ dataset contains 100 training OOD examples. However, we decided not to use them for training due to the nature of our experiments. There are also 100 validation and 1,000 testing OOD examples.

4.2 Evaluation Process

We evaluate the model on the downstream task of OOD data detection and measure the change in

OOD data detection metrics. We designed experiments in the following way. We train the OodGAN on IND training examples as a first step. Next, we generate the OOD examples using the trained model of OodGAN. We generate the same number of OOD examples as a number of IND examples in the training set. In a third step, we train the threshold-based OOD detection model using cross-entropy loss on training IND examples and negative entropy loss on generated OOD examples. In the last step, we evaluate both IND and OOD metrics.

4.3 Metrics

We evaluate the OodGAN by measuring metrics on the downstream task of OOD detection. We measure AUROC, AUPR, and FPR*N* metrics (Ren et al., 2019; Hendrycks and Gimpel, 2017; Hendrycks et al., 2019) to evaluate OodGAN's ability to generate OOD data that helps IC to distinguish IND and OOD input utterances. We treat OOD examples as the positive class.

- **AUROC** The area under the receiver operating characteristic (ROC) curve. The score says the probability that a randomly selected OOD example will have a higher predicted probability of being an OOD than a randomly selected IND example. Higher AUROC score is better.
- **AUPR** The area under the precision-recall curve when OOD inputs are treated as positive samples. AUPR calculates the average precision score for all recall values. Intuitively, the higher the classification threshold we select, the more OOD will be classified as OOD. However, we risk that more IND will be classified as OOD. AUPR expresses this risk. Higher AUPR score is better.
- **FPR*N*** The false-positive rate (FPR) when the true positive rate (TPR) is N%. FPR*N* metric is a practical value in real-world application since it evaluates an OOD detection performance at a particular threshold. Lower FPR*N* means there is a smaller chance of IND examples triggering false alarm (IND getting classified as OOD) when the model's performance on OOD example is N%. We report FPR when TPR is 0.95 and 0.90. Lower FPR*N* score is better.

We consider FPR*N* metric as the most practical value in real-world application since it evaluates an OOD detection performance at a particular threshold. Lower FPR*N* means there is a smaller chance of IND examples triggering false alarm (IND getting classified as OOD) when the model correctly recognizes N% of OOD examples.

We also measure IND accuracy that evaluates generated OOD data's influence on the IC's ability to recognize the intents of IND data correctly.

- **IND accuracy** The percentage of IND data that have assigned correct intent label. We expect that generated OOD examples cannot improve the IC's ability to recognize intent labels for ID. However, generated OOD examples can degrade the IC's ability to recognize IND intents. Thus, we measure the IND accuracy to evaluate whether generated OOD negatively impacts the IC. Higher IND accuracy is better.

4.4 Implementation

We based our implementation on the Github repository[1] of SeqGAN implemented in PyTorch. The generator is one layer GRU recurrent neural network trained using Adam optimizer with a learning rate set to 0.001. Input to the generator is embedded with fastText embeddings (Joulin et al., 2016) trained on Wikipedia. The generator uses negative log-likelihood loss during LM training and policy gradient loss during GAN training. The discriminator is a two-layer bidirectional GRU recurrent neural network with a tanh activation function. Adagrad optimization is used for training the discriminator with a learning rate set to 0.1 and binary cross-entropy loss is optimized. The auxiliary classifier uses the convolutional neural network proposed by Kim (2014), which has filters of size 2, 3, 4, and 5, and for each size, there are 256 filters. We used the LeakyReLU activation function and 0.5 dropout in output dense layers. The auxiliary classifier is trained using the Adam optimizer with a learning rate set to 0.0001 and cross-entropy loss is optimized.

We show the comparison of number of parameters between OodGAN, SeqGAN, and Zheng et al. (2020) in Table 1.

5 Results

5.1 Results on Zheng et al. (2020)

We first conducted experiments to replicate results reported by Zheng et al. (2020) on the OSQ dataset.

[1] https://github.com/suragnair/seqGAN

	# Parameters
Zheng et al. (2020)	7M
SeqGAN (Yu et al., 2017)	800k
OodGAN	2M

Table 1: Number of parameters

OSQ (Larson et al., 2019)	AUROC ↑	AUPR ↑	FPR 0.95 ↓	FPR 0.90 ↓	IND Acc. ↑
Results reported by Zheng et al. (2020)	95.4	98.9	25.0	10.1	93.3
Our implementation of Zheng et al. (2020)	88.79	58.22	36.49	26.87	88.00

Table 2: OOD detection performance on the OSQ dataset with model proposed by Zheng et al. (2020)

We created our implementation according to the paper's description because there is no publicly accessible implementation of their proposed model. We report results in Table 2.

We could not reproduce the number reported by Zheng et al. (2020) even though we implemented the model as was described in the paper. The experiments showed that the denoising auto-encoder is a weak part of the architecture. Its token accuracy of text reconstruction on the validation set was only 0.37%. Thus, the low performance of the auto-encoder is the reason why the generator generates poor quality examples.

5.2 Results on proposed model OodGAN

First, we want to compare OodGAN with baselines. We selected two baselines to evaluate improvements of our proposed OodGAN. Our baselines for the ROSTD dataset is our implementation of Zheng et al. (2020) and the work of Gangal et al. (2019). The baseline for the OSQ dataset is our implementation of Zheng et al. (2020).

Table 3 shows results on ROSTD dataset and Table 4 shows results on OSQ dataset. Results on ROSTD data are promising. They show around 65% relative improvement in FPR 0.95 compared to baseline of our implementation of Zheng et al. (2020) and around 5% absolute improvement in FPR 0.95 compared to baseline of Gangal et al. (2019). For the more challenging OSQ dataset, there is around 28% relative improvement in both FPR 0.95 and FPR 0.90 compared to the baseline.

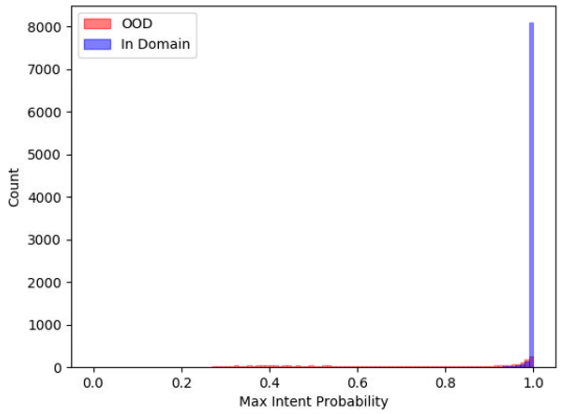
To evaluate whether OodGAN helps the threshold-based OOD detection model to discriminate between OOD and IND examples, we plotted the histogram of the test data's maximum intent probability for system trained with and without generated OOD examples. Figure 3 shows the histogram for ROSTD dataset. Probability scores for IND (blue) and OOD (red) data are spread out over all probability values when there are no OOD data used for model training. Thus it is hard to select

ROSTD (Gangal et al., 2019)	AUROC ↑	AUPR ↑	FPR 0.95 ↓	FPR 0.90 ↓	IND Acc. ↑
w.o. OOD	97.64	93.86	8.10	5.56	**99.05**
Our implementation of Zheng et al. (2020)	88.67	54.84	37.82	26.04	88.00
Gangal et al. (2019)	98.22	**96.47**	7.41	-	-
OodGAN	**98.99**	96.26	**2.59**	**1.37**	98.31

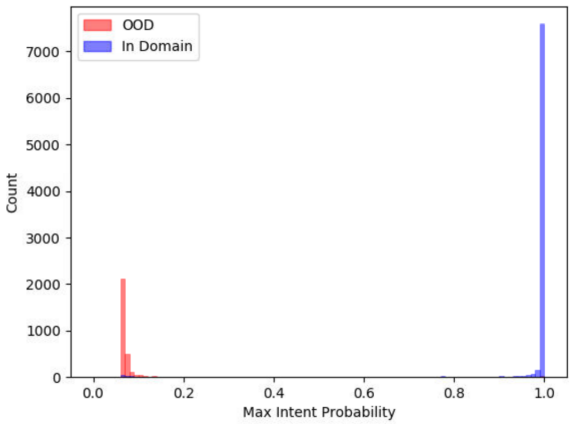
Table 3: OOD detection performance on the ROSTD dataset

OSQ (Larson et al., 2019)	AUROC ↑	AUPR ↑	FPR 0.95 ↓	FPR 0.90 ↓	IND Acc. ↑
w.o. OOD	90.89	**97.99**	28.11	20.98	89.04
Our implementation of Zheng et al. (2020)	88.79	58.22	36.49	26.87	88.00
OodGAN	**91.24**	97.79	**26.07**	**19.29**	**90.11**

Table 4: OOD detection performance on the OSQ dataset



(a) Model trained with no OOD



(b) Model trained with generated OOD

Figure 3: Distributions of detection scores corresponding to the IND and OOD examples of the ROSTD dataset

a well discriminating threshold. The result of the model trained with OOD data is significantly better. The graph shows a clear separation between IND and OOD data, with IND data receiving high intent score and OOD data receiving a low score.

The OOD detection model is combined with IC in many real-world applications. For this reason, the joint accuracy of OOD detection and IND intent recognition is an important metric. We show how the joint accuracy depends on the selected threshold in Figure 4. To draw this graph, we select different thresholds, and we tag examples having an intent score below the threshold as OOD. We classify the intent for the rest. Our proposed approach leads to high joint accuracy of OOD detection and IND intent recognition with low threshold values. That confirms that models trained with generated OOD assign low scores to OOD and high scores to IND examples.

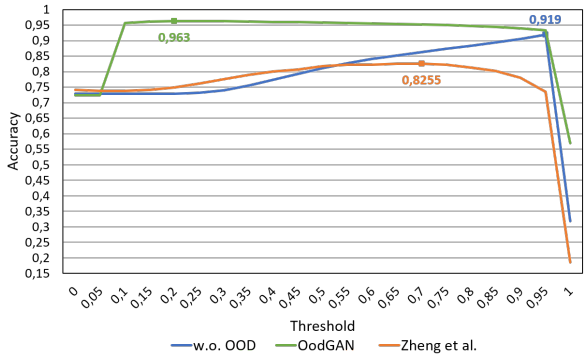


Figure 4: Joint accuracy for ROSTD data across different threshold value. Points mark the highest joint accuracy of OOD detection and IND intent recognition.

The separation between generated OOD examples and IND examples is visible in t-SNE (Hinton and Roweis, 2002) visualization as well. Figure 5 shows the t-SNE visualization of IND and generated OOD data. We can notice that generated data create recognizable clusters close to IND data but do not mix with it. Finally, we list OOD examples generated by OodGAN in table 5.

6 Conclusion

This paper proposed a novel OOD data generation model OodGAN that generates OOD examples that improved OOD detection performance in a dialog system. The model does not require any OOD training examples. Moreover, the model does not rely on the auto-encoder to map utterances into latent space, reducing the model size. It models the data

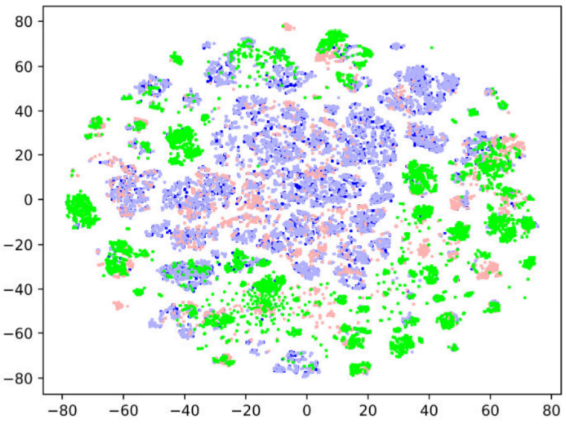


Figure 5: t-SNE visualization of the BERT feature vectors associated with the examples from the ROSTD dataset. IND examples are blue, testing OOD examples are red, and examples generated by OodGAN are green.

IND Examples	Should I be expecting rain today
	I need a new alarm for 8:30 am
	Show my reminders
	Show me the extended forecast please
	Snooze alarm for 5 more minutes
OOD Examples	Why do people watch television
	Where do pineapples grow
	Should I go to the mall today or tomorrow
	Tell me how to install a pool
	Transfer my PayPal balance to my bank
Generated by OodGAN	Remind me of my 4pm and Game of Thrones alarm
	When should I unpack
	Add day at workout please
	Give me my Sarasota appointment
	Do I need to pack to Galway this umbrella

Table 5: Examples sampled from the IND and OOD test set of the ROSTD dataset and OOD utterances generated using OodGAN model.

generator as a stochastic policy in reinforcement learning instead. The model uses two rewards for the generator. The discriminator's reward guides the generator to generate examples as close to the IND data as possible. The auxiliary intent classifier's reward guides the generator to generate examples with low probabilities for all intent classes. Our experiments show that OOD examples generated by OodGAN improve the performance of the OOD detection problem.

References

David Donahue and Anna Rumshisky. 2018. Adversarial text generation without reinforcement learning. *arXiv preprint arXiv:1810.06640*.

Varun Gangal, Abhinav Arora, Arash Einolghozati, and Sonal Gupta. 2019. Likelihood ratios and generative classifiers for unsupervised out-of-domain

detection in task oriented dialog. *arXiv preprint arXiv:1912.12800*.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*.

Dan Hendrycks and Kevin Gimpel. 2017. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *ArXiv*, abs/1610.02136.

Dan Hendrycks, Mantas Mazeika, and Thomas G. Dietterich. 2019. Deep anomaly detection with outlier exposure. *ArXiv*, abs/1812.04606.

Geoffrey E Hinton and Sam Roweis. 2002. Stochastic neighbor embedding. *Advances in neural information processing systems*, 15:857–864.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413.

Stefan Larson, Anish Mahendran, Joseph J Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K Kummerfeld, Kevin Leach, Michael A Laurenzano, Lingjia Tang, et al. 2019. An evaluation dataset for intent classification and out-of-scope prediction. *arXiv preprint arXiv:1909.02027*.

Sungjin Lee and Igor Shalyminov. 2019. Contextual out-of-domain utterance handling with counterfeit data augmentation. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7205–7209. IEEE.

J. Ren, Peter J. Liu, E. Fertig, Jasper Snoek, Ryan Poplin, Mark A. DePristo, Joshua V. Dillon, and Balaji Lakshminarayanan. 2019. Likelihood ratios for out-of-distribution detection. In *NeurIPS*.

Seonghan Ryu, Sangjun Koo, Hwanjo Yu, and Gary Geunbae Lee. 2018. 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.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-first AAAI conference on artificial intelligence*.

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