Tejashri Gadre

tejashri@kale.im

Out-of-Distribution input prediction for Task-Oriented Dialogue Systems

Table of Contents

Tejashri Gadre

tejashri@kale.im

Tejashri Gadre

tejashri@kale.im

Definition

Project Overview

In our day-to-day life, we communicate with the task-oriented dialogue systems (often termed as chat-bots) made available as a part of helpdesk/customer service of online services such as online banking sites, e-ticket booking, e-commerce and many more for resolving our basic queries.
The main purpose of these task-oriented dialogue systems is to assist the users and free up the time of customer service representative for resolving the queries that are complex and could not be resolved by chat-bots. However, often we end up getting directed for such complex queries to a chat-bot, instead of passing these over to the customer service representative. This results in incorrect responses, wasting valuable time of the customer, leading to customer dissatisfaction. This happens because most chat-bot systems are designed in such a way that when out-of-distribution data is provided as input, there is no way to determine if the data is valid enough to pass through the trained model. When such data is fed to the model, it gives the output which is not valid for the given input. As the model has never seen this type of data before, it is bound to give incorrect output, which affects the overall customer experience.

The main objective of this project is to predict the out-of-distribution input data and pass only in-scope data to the chat-bots. The identified out-of- distribution data could be further used for re-training the model. In this way, the risk of giving incorrect output could be minimized.
I tried different approaches to find the solution as below:

- Variational Autoencoder
- Isolation Forest Algorithm
- BERT with SNGP

The first two approaches failed to perform. I have explained what I tried in these approaches and how it helped me to better understand the problem statement and to come to the final solution.

Inspiration

Machine learning/ AI is unable to gain confidence of many industries due to lack of understanding of how it works. When a model trained by supervised learning algorithm, with well-defined target label set, meets real-world, it may perform poorly as the real-word input data does not belong to that

Tejashri Gadre

tejashri@kale.im

"well-defined" label set. This becomes a serious issue when it comes to safety critical systems such as self-driving cars and medical diagnosis.

Identifying out-of-distribution data is important to make current machine learning/AI based systems safe, robust and trustworthy.

Current state

The current implementation as in *An Evaluation Dataset for Intent Classification and Out-of-Scope Prediction* by Larson et al., which was published in EMNLP in 2019. The GitHub page for this dataset is linked here. It tries 3 techniques to address the problem.

1. Adding out-of-scope data to training data
2. Setting a threshold value on prediction probability for each class
3. Using binary classes (0 and 1) to classify in-scope and out-of-scope data

It can be seen that none of these methods were able to increase the recall value for out-of-scope data. (Larson et al., 2019)

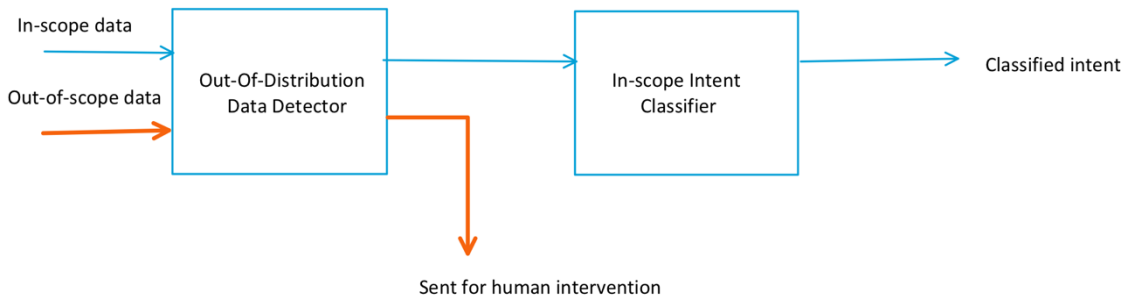
Problem statement

To predict the out-of-distribution input samples, that do not fall in scope of trained model, before passing these for prediction, thereby minimizing the risk of incorrect classification.

It is possible to predict out-of-distribution data (OOD) for input data with following various available techniques, such as using standard deviation, plotting box plots, DBScan clustering. However, none of these give the optimal solution.

The model is trained on in-distribution data. The Out-of-distribution data is never seen by model.

I intent to design a model which detects OOD data before passing it the actual classification model. This model detecting OOD data will be trained using unsupervised learning methods.



Tejashri Gadre
tejashri@kale.im

Metrics

‘Precision, Recall and AUPRC (Area Under Precision Recall Curve) is used to evaluate out-of-distribution prediction. AUPRC is suitable where you care more about finding positive examples(In this case OOD data).

In this project , the intension is to find all OOD data (perfect recall) without accidently classifying any OOD data as in-distribution data (perfect precision). Thus we will be using these 3 metrics.

Tejashri Gadre

tejashri@kale.im

Analysis

Data Exploration

Kaggle dataset is be used to implement this project.

Kaggle Dataset : Out-Of-Scope Intent Classification Dataset

(https://www.kaggle.com/stefanlarson/outofscope-intent-classification-dataset)

The input files are in. json format.

Data has two features question and intent. The question feature defines various queries asked to chat-bots and the intent feature defines intent of that query.

For example, for query “What is my credit card blocked”, intent is defined as “account-blocked”

is_train.json, is_test.json, is_val.json : These files contain train/test/val data for in-scope data points. Teach input data is divided into two parts namely question and intent. There are 150 distinct intent values which are the classes. The intent value serves as target label. All classes are balanced with 100 entries for each class. Only is_train.json file data used to train the model.

oos_train.json, oos_test.json, oos_val.json : These files contain train/test/val data for out-of-scope data points. This data would be used to test the model trained on in-scope data to verify if it is able to identify the OOD data. Only oos_test.json fie data is used for model evaluation which contains 1000 oos queries.

Exploratory Visualization

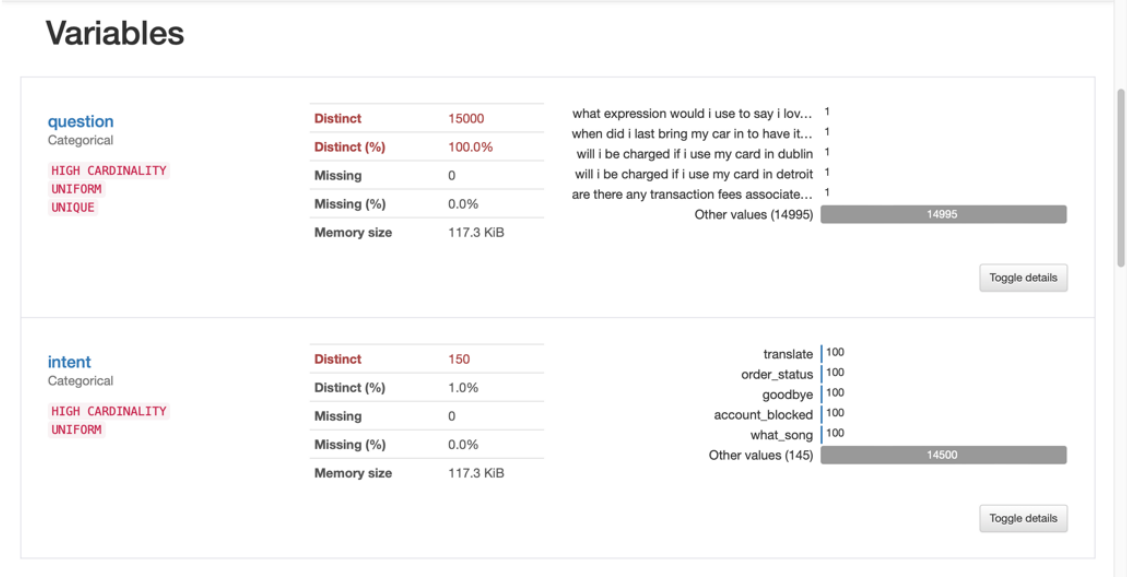
Data from is_train.json is used for training and data from oos_test.json is used for evaluation. The json file is loaded in pandas dataframe and two columns are added as ‘Question’ and ‘intent’ to store query and intent present in the json file.

The visualization below is created using Pandas profiler report to show balanced distribution of training data across all classes.

Tejashri Gadre

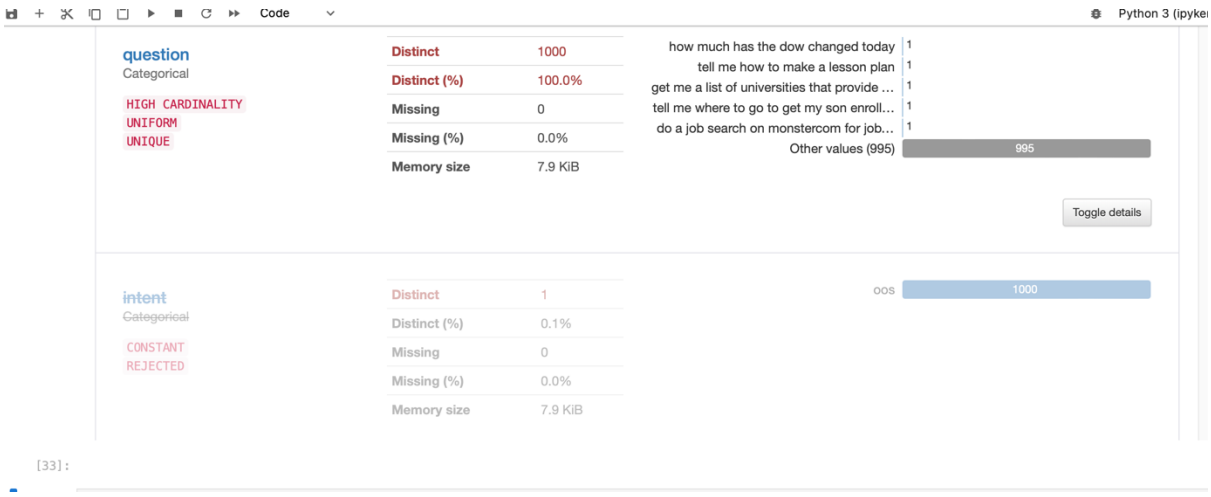
tejashri@kale.im

Train data



Evaluation data

(There is only one intent label in this data, 'oos', to indicate that these are out-of-distribution samples.)



Ngrams

Ngrams sorted based on count are plotted to see frequently occurring bigrams and trigrams for questions in train dataset.

trigrams = get_ngram_count(df_train['question'],10,(3,3)).sort_values(by='count')

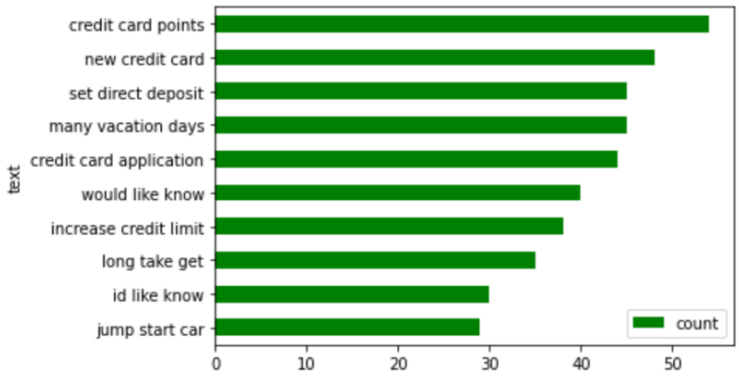
bigrams = get_ngram_count(df_train['question'],10,(2,2)).sort_values(by='count')

Tejashri Gadre

tejashri@kale.im

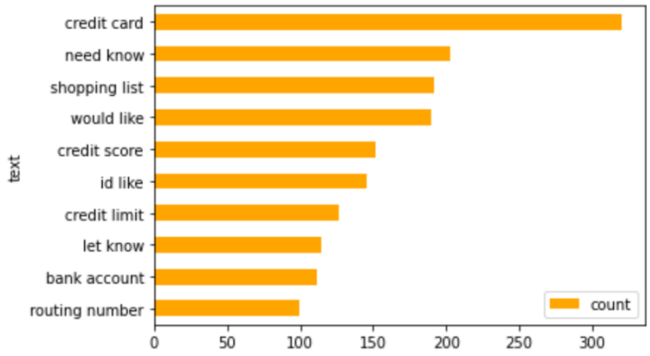
```
trigrams.plot.barh(x='text', y ='count', color='green')
```

```
<AxesSubplot:ylabel='text'>
```




```
ax = bigrams.plot.barh(x='text', y ='count', color='orange')
ax
```

```
<AxesSubplot:ylabel='text'>
```



Features – 'Question'

Below are the first few rows from training dataset.

First rows

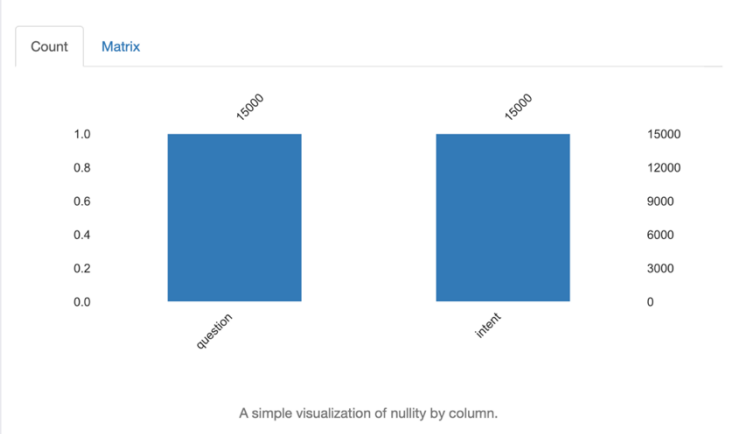
	question	intent
0	what expression would i use to say i love you if i were an italian	translate
1	can you tell me how to say 'i do not speak much spanish', in spanish	translate
2	what is the equivalent of, 'life is good' in french	translate
3	tell me how to say, 'it is a beautiful morning' in italian	translate
4	if i were mongolian, how would i say that i am a tourist	translate
5	how do i say 'hotel' in finnish	translate
6	i need you to translate the sentence, 'we will be there soon' into portuguese	translate
7	please tell me how to ask for a taxi in french	translate
8	can you tell me how i would say, 'more bread please' in french	translate
9	what is the correct way to say 'i am a visitor' in french	translate

Tejashri Gadre

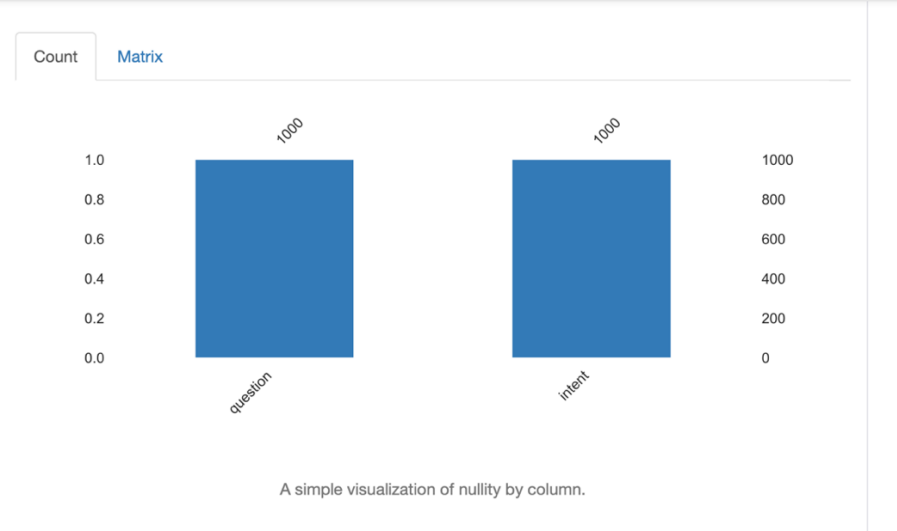
tejashri@kale.im

Maximum question length for training dataset is 28. The questions include stop words and punctuations which were removed in data preprocessing step. There are no null values for both the features in training and evaluation dataset.

Training data



Evaluation data



Algorithms & techniques

In this project, I tried below approaches to predict OOD data.

Variational Autoencoders (https://arxiv.org/pdf/1511.06349.pdf)

Variational autoencoders have different applications such as dimensionality reduction(compression), de-noising images, sample generation (generating image or text data) and out-of-distribution data detection.

Tejashri Gadre

tejashri@kale.im

I designed a VAE for regenerating input data, VAE model is trained on in-distribution data. Input is passed though VAE model and is regenerated at the output end. The accuracy of the model will be how well it can regenerate the in-distribution data and when OOD data is passed to this model it will fail to regenerate the OOD data.

Isolation Forest

(https://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/icdm08b.pdf)

This is an unsupervised algorithm. It builds an ensemble of iTrees for given input data set, then anomalies or OOD are those data points which have short average path lengths on iTrees. (Fei Tony Liu et al., 2008)

In this approach, 'Isolation Forest' will serve as the 'Out-of-Distribution Data Detector' and would detect OOD data as anomaly.

Spectral-normalized Neural Gaussian Process (SNGP) with BERT

(https://arxiv.org/abs/2006.10108)

In this approach, BERT classifier is used with SNGP to output confidence probabilities for each intent along with their classification. Since OOD data is never seen by the model generates low confidence on OOD input.

Feature usage

1. VAE Model - Only 'question' feature will be used to train the model, 'intent' feature will not be used for training
2. Isolation Forest Model - Only 'question' feature will be used to train the model, 'intent' feature will not be used for training
3. BERT with SNGP Model – Both 'question' and 'intent' features will be used for training.

Benchmark Model

I have used results provided by dataset authors (Larson et al., 2019) on oos-threshold technique as the benchmark model, as oos-train and oos-binary techniques have used OOD data to train the OOD predictor. Out of the above three techniques used in this paper I have only focused on oos-threshold with BERT as the classifier (highlighted in Yellow below in results section) as I do not think that OOD data should be used to train the intent classifier.

Tejashri Gadre

tejashri@kale.im

Given the implicit lack of availability of OOD data for training in most real-world scenarios I have not used OOD data for training the intent classifier. I have used BERT as the intent classifier trained on in-distribution data and apply OOD data detector in front/after this model to filter out OOD data for human review and pass only in-distribution data to the classifier.

		In-Scope Accuracy				Out-Of-Scope Recall			
	Classifier	**Full**	**Small**	**Imbal**	**OOS+**	**Full**	**Small**	**Imbal**	**OOS+**
oos-train	FastText	89.0	84.5	87.2	89.2	9.7	23.2	12.2	32.2
	SVM	91.0	89.6	89.9	90.1	14.5	18.6	16.0	29.8
	CNN	91.2	88.9	89.1	91.0	18.9	22.2	19.0	34.2
	DialogFlow	91.7	89.4	90.7	91.7	14.0	14.1	15.3	28.5
	Rasa	91.5	88.9	89.2	90.9	45.3	**55.0**	**49.6**	**66.0**
	MLP	93.5	91.5	92.5	94.1	**47.4**	52.2	35.6	53.9
	BERT	**96.9**	**96.4**	**96.3**	**96.7**	40.3	40.9	43.8	59.2
oos-threshold	SVM	88.2	85.6	86.0	—	18.0	13.0	0.0	—
	FastText	88.6	84.8	86.6	—	28.3	6.0	33.2	—
	DialogFlow	90.8	89.2	89.2	—	26.7	20.5	38.1	—
	Rasa	90.9	89.6	89.4	—	31.2	1.0	0.0	—
	CNN	90.9	88.9	90.0	—	30.9	25.5	26.9	—
	MLP	93.4	91.3	92.5		49.1	32.4	13.3	
	BERT	**96.2**	**96.2**	**95.9**	—	**52.3**	**58.9**	**52.8**	—

Methodology

Data pre-processing

Following pre-processing steps were performed.

1. Removed stop words and punctuations from dataset

   The input questions had many stop words such as a, an, the, and, before, after which were removed from both training and test dataset. Punctuations were also removed which were not necessary. This step made the data more relevant and concise.

2. Tokenization and embedding vector creation

   As machine learning models understand numbers, passing text data directly will not work. For Variational Autoencoder, sentence embeddings were created from input sentences. Keras preprocessing text library was used for tokenization and GloVe word embeddings were used to create embedding vectors.

Tejashri Gadre

tejashri@kale.im

For Isolation Forest algorithm , I have used tensorflow Universal Sentence Encoder to generate sentence embeddings for preprocessing Isolation Forest input data.

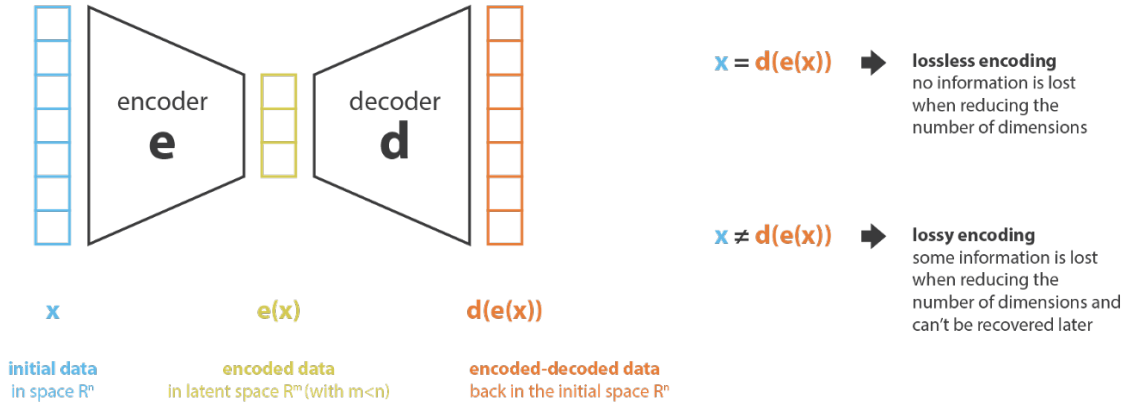
For SNGP-BERT model, input sentences were directly passed by converting pandas dataframes to tensorflow tensors and updating integer encodings for labels.

Tejashri Gadre

tejashri@kale.im

Implementation & refinement

Approach 1 - Using Variational Autoencoder



(https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73)
Variational autoencoder model is implemented to detect out-of-distribution data using reconstruction approach. There are two components in this model the encoder and the decoder. The input data is passed to encoder that learns a compressed(lower-dimensional) representation of input; this compressed form of data is then passed to decoder which tries to regenerate the input data with the original dimensions.

When VAE is trained on large dataset, encoder learns a function to generate two vectors in latent space for mean and variance respectively. From this distribution the latent vector(m) is sampled which is then transformed back to original vector(n) by the decoder, where is m<n.

When out-of-distribution data that model has never seen before, is passed to VAE model, it generates the output with extreme reconstruction loss. However, I expected that when in-distribution data is passed to model, it will generate output with very small reconstruction loss. Thus, based on this loss value model can classify OOD data.

Why LSTM is used?

Traditional neural networks suffer from short-term memory and vanishing gradient problems. LSTMs are good at memorizing information. Other text classification techniques work at word level where as LSTM can work with multiple words and it can process entire input string. In given problem input is a sequence of multiple words . Thus, LSTM will perform better than RNN network.

Tejashri Gadre

tejashri@kale.im

(https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/, https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/)

Input

- All the tokenized input sequences are padded to maximum length of 8 and 50d-glove embedding vector is generated for each sentence of the input dataset.
- The embedding vector of each sentence is of shape (8, 50).
- Latent vector dimension m is selected such that m < n where n is the dimension of original input vector. Latent dimension is set to 4 to ensure that bottleneck is created and input data is compressed to extract important features.

Encoder

- Encoder is created with Input Layer, LSTM layer, Sampling Layer and Output Layer. The original input vector(n) is passed as input to encoder.
- Input Layer : Sentence embeddings with shape (batch_size, 8, 50) are passed to this layer.
- LSTM layer : LSTM layer with 128 neurons is created. Input layer embeddings are passed to this layer as input. return_sequences is set to False as only hidden state output and cell state for the last input time step is required. Output shape is (batch_size, 128).
- Sampling Layer : Output of LSTM layer is passed to sampling function which then calculated the mean and variance of the input and passed it to sampling layer. The output of this layer is the latent vector with reduced dimension of (batch_size, 4).

```
Model: "encoder"
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input_2 (InputLayer)        [(None, 8, 50)]           0         []

 lstm_2 (LSTM)               (None, 256)               314368    ['input_2[0][0]']

 dropout_2 (Dropout)         (None, 256)               0         ['lstm_2[0][0]']

 x (Dense)                   (None, 256)               65792     ['dropout_2[0][0]']

 dropout_3 (Dropout)         (None, 256)               0         ['x[0][0]']

 z_mean (Dense)              (None, 4)                 1028      ['dropout_3[0][0]']

 z_log_var (Dense)           (None, 4)                 1028      ['dropout_3[0][0]']

 sampling_1 (Sampling)       (None, 4)                 0         ['z_mean[0][0]',
                                                                  'z_log_var[0][0]']

==================================================================================================
Total params: 382,216
Trainable params: 382,216
Non-trainable params: 0
```

Tejashri Gadre

tejashri@kale.im

Decoder

Decoder is created to reconstruct the original input vector from provided latent vector. Below layers are added to decoder.

- Input layer : Latent input vector generated by encoder is used as input to the decoder .
- Repeated Vector : RepeatedVector() is used to repeat the input vector to get the original sequence length.
- LSTM layer : LSTM layer with 128 neurons is created and return_sequences is set to True as here hidden state output for each input time step is returned .
- TimeDistributed Dense Layer : It is wrapper layer. It applies same dense operation to every timestep of previous layer output.
- output Layer : A dense layer is added as output layer .

```
Model: "decoder"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 z_sampling (InputLayer)     [(None, 4)]               0

 repeat_vector_1 (RepeatVect  (None, 8, 4)             0
 or)

 lstm_3 (LSTM)               (None, 8, 256)            267264

 time_distributed_1 (TimeDis  (None, 8, 50)            12850
 tributed)

 dense_3 (Dense)             (None, 8, 50)             2550

=================================================================
Total params: 282,664
Trainable params: 282,664
Non-trainable params: 0
_________________________________________________________________
```

Validation

The output of the decoder is then compared with the input vector. If both are same that means model was able to reconstruct the input and the passed input data is not out-of-distribution data. On the other hand, if decoder fails to reconstruct the input and there is significant difference in input and output that means the input data passed to model is out-of-distribution data as model has never seen such data before.

Loss Function

The loss function for VAE is combination of reconstruction loss and KL loss. (https://keras.io/examples/generative/vae/)

Tejashri Gadre

tejashri@kale.im

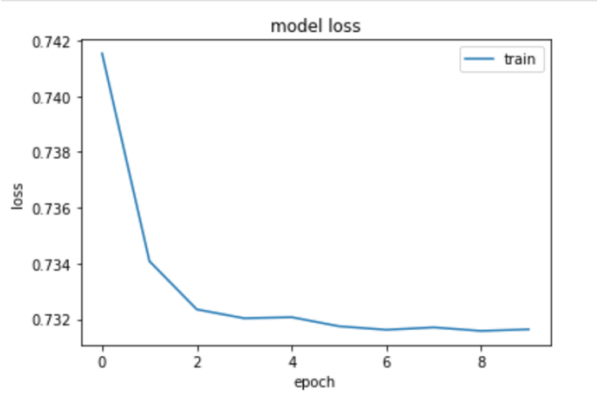
For reconstruction loss , I have used cosine_similarity function. We will be calculating loss by comparing input embedding vector with generated output embedding vector. cosine_similarity(Input vector, output vector) return a value which shows how similar two vectors are where -1 = most similar.

KL loss is added to the reconstruction loss, where mean and variance are used to calculate loss. This loss component ensures that the latent space generated is uniform.

$$\sum_{i=1}^{n} \sigma_i^2 + \mu_i^2 - \log(\sigma_i) - 1$$

(https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf)

Total loss = reconstruction loss + KL loss



Challenges

- There was no existing model for Text regeneration VAE. I referred to Image regeneration VAE while creating this model.
- I created feature vectors for each input sequence and passed it to the model expecting the same feature vector will be regenerated at the output. However, when I compared input and output vectors with cosine similarity function there was a lot of dissimilarity in the vectors.
- The training was not successful as there was only 30 % accuracy.
- This approach was not working so I tried changing the reconstruction loss function to binary cross entropy as I found many VAE models were trained using this function. After using this

Tejashri Gadre

tejashri@kale.im

loss function model was trained better with minimal loss. However, when I compared training dataset input vector with output vector, they were still dissimilar. The model was getting trained, but it was not regenerating the output. Thus, binary cross entropy function was not the right choice.

- To address this issue, I tried to tune the network architecture, learning rate, activation function. But none of these changes helped to improve performance.

Future works

- I think the problem lies with the feature extraction step where sentences are converted into embedding vectors and selection of loss function. Sentence generation due to dependency on time steps is difficult to train with and would require careful feature extraction and network design. If I could do better feature extraction it might improve the model performance.
- Due to limited time constraints I could not try more approaches. But I think VAE can definitely be used for OOD data detection for text data as OOD detection for images has already been implemented successfully. I will keep working on this model.

Approach 2 – Using Isolation Forest

Isolation forests are ensemble of binary decision trees. When input data is passed to this algorithm, it generates isolation trees.
(https://www.analyticsvidhya.com/blog/2021/07/anomaly-detection-using-isolation-forest-a-complete-guide/)

Input
Universal sentence encoder is used to generate embeddings for the input data which is then passed to isolation forest algorithm.

Steps

- From given input dataset random sub-sample of the data is selected and assigned to a binary tree.
- Branching is done based at random threshold which is any value in the range of minimum and maximum value of the feature.
- If value of the data point is less than the selected threshold, it is assigned to the left branch else to the right.

Tejashri Gadre

tejashri@kale.im

- The process is repeated recursively till each data point completely isolated or maximum depth is reached.

After ensemble iTrees (Isolation Trees) are created model training is completed.
For validation, each input is traversed through all trees of the model. An anomaly score is assigned to each input based on depth of the tree traversed to arrive at that input. -1 score is assigned to anomalies that is out-of-distribution and 1 to in-distribution input.

Challenges

- After training on in-distribution data with zero contamination, isolation forest algorithm classified all out-of-distribution data as in-liars. Given the nature of the data, isolation forest with contamination set as auto, classified 10% of out-of-distribution data as anomaly, but in-distribution sentences with similar sentence structures were also classified as anomaly.
- Tendency of this algorithm to be influenced by sentence structures led me to search for a solution that would rely more on confidence probabilities of each intent of the intent classifier, rather pursuing approaches that kept in-distribution intents separated from out-of-distribution data detection.
- The algorithm needed intents to classify correctly. However, when model was trained using intents, at the time of validation I did not have any intent for OOD data, and the model was not accepting input without intent feature.

Imbalanced fit data (900 in-distribution + 100 out-of-distribution)	With stop words removed	With stop words
True positives	73.9%	61.5%
False negatives	76%	54%

Future works

For this approach to be effective, I would need to work more on feature extraction. Attempts to fit with in-distribution data failed to detect out-of-distribution data, whereas training with limited out-of-distribution data resulted in too many false negatives to be useful.

Tejashri Gadre

tejashri@kale.im

Approach 3 – Using BERT with SNGP

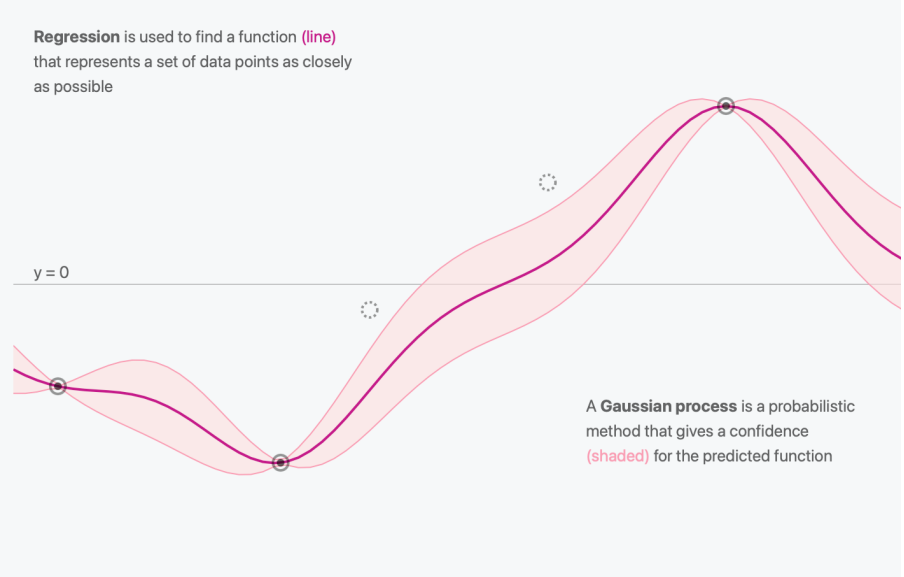
In this approach I have used BERT model as it is pre-trained on a large corpus of text data. It computes vector space representations of natural language that are suitable for learning.
I have fine-tuned this model for OOD detection with Spectral-normalized Neural Gaussian Process (SNGP).
(https://www.tensorflow.org/tutorials/understanding/sngp)

What is SNGP?

SNGP is used to improve classifier's distance awareness by applying simple modifications to the existing network. A model's distance awareness is a measure of how its predictive probability reflects the distance between the test example and the training data. It applies spectral normalization to hidden layers and dense layer is replaced by a Gaussian process.
(https://arxiv.org/abs/2006.10108)

What is Gaussian process?

A Gaussian distribution or a normal distribution defined mean & a covariance matrix is the basic building block of a Gaussian process. Simply put, a Gaussian process is a set of random variables. In the following diagram, while regression is used to fit a line through a set of data points, a Gaussian process provides the confidence on the predicted function. This attribute makes them attractive for use in out-of-distribution (OOD) detection without training with OOD data.



(https://distill.pub/2019/visual-exploration-gaussian-processes/)

Tejashri Gadre

tejashri@kale.im

What is Spectral Normalization?

Spectral normalization is a weight normalization technique to address instability of training generative neural networks, which often face problems of exploding and vanishing gradients. I observed this problem during 'Approach 1 – Using Variational Autoencoder' for OOD detection. (https://arxiv.org/abs/1802.05957)

Input

I converted train & test data loaded with pandas dataframe to tensorflow tensors and added unique encoding (0-149) for intent labels by combining train and test data sample set. I have prepared OOD_eval_dataset by concatenating in-distribution test and out-of-distribution test data. This data was passed to SNGPBertClassifier.
(https://www.tensorflow.org/text/tutorials/uncertainty_quantification_with_sngp_bert)

Steps

1. SNGPBertClassifier definition published by Google on GitHub, which enhances BERT pre-trained model to also emit confidence probabilities using Spectral Normalization & Gausian Process, is used to create a model instance
2. Standard Bert Optimizer is used with Adam Weight Decay & Polynomial Decay schedules to slowly increase the learning rate during the training process
3. Sparse Categorical Cross-entropy loss was chosen as this is a classification problem with mutually exclusive intents with integer representation
4. I tried to run a few variations to test model performance with epoch sizes as 2, 3 & 4 as well as batch sizes of 16 & 32 with AWS SageMaker Studio using 'ml.g4dn.xlarge' GPU enabled instance and Tensorflow optimized for GPU execution, given that normal execution time on my laptop turned out to be 6 hours for 3 epochs.

Unlike, other 2 approaches, this approach was successful in classifying in-distribution data and detecting out-of-distribution samples effectively and the results are discussed in the following sections.

Tejashri Gadre

tejashri@kale.im

Results & visualizations

Model Evaluation & Validation

Metrics used

Precision

Precision is a measure of how well a model avoids classifying negative examples as positive. It is calculated as (True Positives/ (True Positives + False Positives))

In this project, precision is the measure of in-distribution examples not being classified as Out-of-distribution. A higher number of false positives would mean lower precision.

Recall

Recall is a measure of how well a model avoids classifying positive examples as negative. It is calculated as (True Positives / (True Positives + False Negatives)).

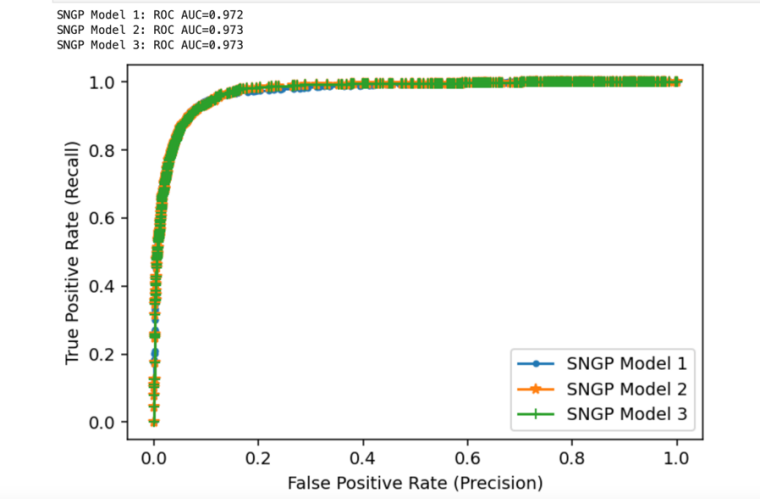
In this project, recall is the measure of out-of-distribution examples not being classified as in-distribution. A higher number of false negatives would mean lower recall.

AUPRC (Area Under Precision and Recall Curve)

AUPRC metric is used with imbalanced data where the focus is optimizing for true positives. Maximum AUPRC means perfect precision and perfect recall. In real-world it is always a trade-off between precision and recall.

In this project, objective is to find out-of-distribution data efficiently. Thus, recall is to be optimized.

Below graph shows AUPRC curve for 3 runs.



Tejashri Gadre

tejashri@kale.im

Metric	SNGP Model 1 Batch size = 32 Epochs = 3	SNGP Model 2 Batch size = 16 Epochs = 2	SNGP Model 3 Batch size = 16 Epochs = 4
Training Accuracy	98.53%	99.39%	99.75%
Validation Accuracy	95.98%	95.98%	96.27%
AUPRC	90.26%	89.26%	89.15%

Justification

Metric	Benchmark Model BERT – Full	SNGP Model 3 Batch size = 16 Epochs = 4
Validation Accuracy	96.2 %	96.29%
Recall	52.3%	100%

Conclusion

Reflection

In safety critical applications and real-world scenarios, it is essential to detect out-of-distribution data for AI models to be effective. In this project, I learnt various aspects of solving OOD detection problem and faced challenges that triggered me to focus on characteristics of data being used to train the model.

This was my first NLP project and it helped me understand concepts like word embeddings, RNNs and GANs in a more hands-on way. I always thought images are the more complex to perceive than text but now I have realized that it is equally challenging to design NLP AI systems.

After trying to implement multiple approaches to tackle this problem I have understood the complexity that it presents, and I am eager to explore more in this field. This problem statement is relevant across domains including education and healthcare that interest me the most and this project will serve as my first step.

Tejashri Gadre

tejashri@kale.im

Improvements

I would like to learn more about Spectral Normalization and Gaussian processes and try to optimize SNGP BERT Classifier. I think hyperparameter tuning could further enhance the model performance and I will try to use AWS SageMaker hyperparameter tuning job to find best hyperparameters for this problem.

I would like to further explore Variational Autoencoders for text OOD detection similar to how effectively these are being used with image data.

Works Cited

Larson et al. (2019). An Evaluation Dataset for Intent Classification and Out-of-Scope Prediction. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 1311–1316, Hong Kong, China, November 3–7, 2019. c 2019 Association for Computational L.

Fei Tony Liu et al. (2008). *Isolation Forest*. (IEEE) Retrieved from https://ieeexplore.ieee.org/document/4781136

Feng et al. (2021). Improving Variational Autoencoder based Out-of-Distribution Detection for Embedded Real-time Applications. Retrieved from https://arxiv.org/pdf/2107.11750.pdf

Jeremiah Zhe Liu et al. (2020). Simple and Principled Uncertainty Estimation with Deterministic Deep Learning via Distance Awareness. Retrieved from https://arxiv.org/abs/2006.10108