Knowing What You Do Not Know: Investigating Large Language Models for Out-of-Domain Intent Classification

Stanford CS224N Custom Project

Claire Tang
Department of Computer Science
Stanford University
cltang@stanford.edu

Abstract

Intent classification is an important natural language understanding task which involves identifying all the supported intents, referred to as in-domain, from user utterances while also detecting and rejecting out-of-domain, or OOD, user intents. The purpose of this project is to investigate how effective pre-trained large language models such as BERT or GPT are for intent classification when fine-tuned on specific in-domain datasets. We implemented and evaluated minBERT pretrained and fine-tuned models on CLINC dataset [1]. We implemented OOD detection methods based on [1, 2, 3, 4]. We also fine-tuned multiple large language models from HunggingFace, both encoder-based and decoder-based, and evaluated their performance based on the same CLINC dataset. We present detailed findings from our extensive evaluations.

1 Key Information to include

- Mentor: Irena Gao

2 Introduction

With the increasing popularity of intelligent devices like chatbot and voice-enabled personal assistants, understanding the intent behind the user utterance, a.k.a *intent classification*, becomes increasingly important. Moreover, these devices are increasingly being used in open environments with personal or new vocabulary, and even new intents that are out of the original model design scope. Thus, it becomes more and more challenging to accurately understand the user intents. On one hand, it needs to ensure high accuracy when classifying the supported intents (**in-domain (IND)**), and on the other hand, it needs to be able to correctly identify and reject (**out-of-domain (OOD)**) intent. Basically it requires the model to know the boundary between *what it knows* and *what it does not know*. If the models cannot detect OOD effectively, the consequences can range from giving wrong answer to a non harmful question (in a socialbot), to giving factually wrong results to user search query (eg., based on ChatGPT due to its hallucination), to more serious mistakes like allowing intelligent Robot to perform physical attack to human (due to unwanted arm movement). It is highly desirable to design natural language understanding models such as intent classifiers that can perform well for both in-domain prediction and OOD detection.

Recently we are witnessing the explosive development of large language models, from BERT [5] to GPT[6]. These foundational models are trained on huge amount of data. Because of that, these pre-trained large models have demonstrated to be able to greatly improve task-agnostic few-shot performance. They allow quick generalization of machine learning models. However, these models are focused on task agnostic performance. When used for a specific task, fine-tuning is required

which is a process that updates the weights of a pre-trained model by training on a new labelled dataset for your specific task, for example, an intent classifier for a particular domain.

In this project we are interested in studying when large pre-trained models are used to fine-tune intent classifier for specific tasks, how well these models can help addressing the OOD problem.

3 Related Work

Intent Classification is one of the most important natural language understanding tasks [7]. Various models have been proposed to encode the user utterance for intent classification, from RNN [8], enriched word embeddings [9] to BERT based model [5].

OOD Detection has been studied for many years since [10]. Authors in [11] built an SVM classifier trained on in-distribution (IND) data and randomly selected OOD data. More recently [12] have also studied this problem in Question-Answering systems.

On the other hand, some researchers in computer vision have created techniques that rely on either calibrating the confidence of predicted classes (ODIN) [2] or constructing a Gaussian model using features extracted from neural network hidden layers (Mahalanobis) [3]. Monto-Carlo Dropout (MC dropout) [4] approximates Bayesian inference to learn deep learning model uncertainty and its recent investigation in Transformer models [13].

From dataset point of view, CLINC [1] is a benchmark dataset that allows intent classifiers to be tested on both IND prediction and OOD detection performance. For OOD performance, there are different evaluation metrics that have appeared in literature [2, 3, 14].

We implemented these methods and evaluation metrics in this project. We will provide many more details in later sections for some of the above mentioned related work.

4 Approach & Implementation

We have finished the following main implementation tasks: (1) minBERT function implementation as described in default final project, even though this is not strictly related to our project; (2) multiple out-of-domain detection methods and evaluation metrics implementation based on literature [1, 2, 3, 4]. (3) Hugging Face implementation that leverages Hugging Face APIs (e.g., Tokenizer, Trainer, etc.) to enable the training and evaluation (both in-domain and out-of-domain evaluation) of various Hugging Face models. We have also fine tuned multiple encoder and decoder models from HuggingFace and evaluated their performance on CLINC datasets [1].

minBERT: Following the default final project instructions, we implemented the below functions:

- `bert.py` including attention function inside BertSelfAttention, BertLayer class, BertModel embed class. We refer to the Hugging Face code `https://github.com/huggingface/transformers/blob/main/src/transformers/models/bert/modeling_bert.py` for our own implementations. We passed the sanity_check.py.
- `optimizer.py` including the step function for Adam optimizer class. We refer to the PyTorch source code `https://github.com/pytorch/pytorch/blob/master/torch/optim/adam.py` for our own implementation. We passed the optimizer_test.py.
- `classifier.py` including BertSentimentClassifier class.

We have trained both minBERT-pretrain model and minBERT-finetune model with hyperparamters described in section 5 experimental details. Even though it is not within the scope of our project, as further sanity check, we are able to reproduce the sentiment classification on both the SST and the CFIMDB datasets.

Hugging Face Model Implementation: First, we implemented sequence classification task training and testing code in `huggingfaceSequenceClassification.py` using Hugging Face functions including tokenizer, trainer and model APIs. Our implementation is generic for sequence classification tasks based on different language models. Specifically, `huggingfaceSequenceClassification.py` reads the config files `encoder_models.config` and `decoder_models.config`. In addition to in-domain test, we also implemented out-of-domain

detection test code via our implemented OOD detection method and evaluation code in `ood/` folder which we will introduce in the rest of this section.

Moreover, for GPT-2 generation model, we use the `run_clm.py` in Hugging Face to finetune the model and we implemented `gpt2_inference.py` to generate text using the finetuned GPT-2 model.

For GPT-3 generation model finetuning and inference, we created OpenAI account and used its APIs in `https://platform.openai.com/docs/guides/fine-tuning`.

Out-of-domain Detection Methods: First, we implemented the following deterministic OOD detection methods. For each data sample x, we implemented the following methods in `ood/ood_detection.py` for Hugging Face models as well as the corresponding `ood/ood_detection_minbert.py` that is compatible with minBERT implementation. (l is the output logits in the last layer of neural networks, and f is the feature in the penultimate layer before the last linear classifier layer.

- *Confidence Score:* $\textit{Confidence}(x) = \max[\textit{softmax}(l(x))]$
- *Entropy Confidence Score:* $\textit{Entropy}(x) = \textit{Entropy}[\textit{softmax}(l(x))]$
- *ODIN Confidence Score [2]:* $\textit{ODIN}(x) = \max[\textit{softmax}(l(x)/T)]$ with large temperature scale T (In our experiment, we evaluate on different $T = 10, 100, 1000, 1000$).
- *Mahalanobis distance [3]:* $S_{Maha}(x) = \max_i -(f(x) - \mu_c)^T \Sigma^{-1} (f(x) - \mu_c)$ where μ_c and Σ are the class mean and the covariance matrix.

$$\mu_c = \frac{1}{N_c} \sum_{i:y_i=c} f(x_i), \quad \Sigma = \frac{1}{N} \sum_c \sum_{i:y_i=c} (f(x_i) - \mu_c)(f(x_i) - \mu_c)^T$$

  We refer to the following code `https://github.com/pokaxpoka/deep_Mahalanobis_detector` for our implementation.

We also implemented Monte-Carlo Dropout (MC Dropout) with the dropout rate p [4, 13]. Let x_h be the outputs of the h^{th} layer of the network. MC dropout considers a dropout mask M_h drawn from the Bernoulli distribution:

$$x_h = \sigma(x_{h-1}|W_h, M_h), \quad M_h \sim Bernoulli(1-p)$$

MC dropout can be used at both training and inference stages for each dropout layer of the network and subsequently provides an ensemble of models parameterized by these dropout masks.

We use the following OOD detection scores/methods on T stochastic passes of MC dropout models.

- *Sampled maximum probability (MaxProbMC)*: $\max \textit{Average}[\textit{softmax}(l(x))]$
- *ODIN1000MC*: $\max \textit{Average}[\textit{softmax}(l(x)/T)]$ with temperature scale $T = 1000$
- *Entropy*: $\textit{Entropy}[\textit{Average}[\textit{softmax}(l(x))]]$

We refer to the following code `https://github.com/s-nlp/certain-transformer` for our implementation.

Out-of-domain Evaluation Metrics: Next, we implemented the out-of-domain detection evaluation metrics commonly used in [2, 3, 14]. Our implementation is in `ood/ood_metrics.py`.

Let TP, TN, FP, and FN denote true positive, true negative, false positive, and false negative.

- *EER (Equal Error Rate)*: is the error rate when false positive rate (FPR=FP/(FP+TN)) is equal to the false negative rate (FNR=FN/(TP+FN)). ERR is the lower the better)
- *FPR95*: is the FPR (the probability that an OOD utterance is misclassified as in-domain) when the true positive rate (TPR=TP/(TP+FN)) is as high as 95%. FPR95 is the lower the better.
- *Detection Error*: is the minimum misclassification probability with different thresholds. It is also the lower the better. Based on [2], it is defined as follows:

$$\min_{\delta} \{\delta(1 - TPR) + (1 - \delta)FPR\}$$

- *AUROC* is the Area under the Receiver Operating Characteristic Curve which plots TPR against the FPR with different thresholds. It is the higher the better.
- *AUPR*: is the Area under the Precision-Recall Curve which plots the precision (TP/(TP+FP)) against recall (TP/(TP+FN)) with different thresholds. It is the higher the better. AUPR IN and AUPR OUT are when AUPR is used for IND and OOD data samples respectively.

EER, AUROC, and AUPR IN, AUPR OUT are all threshold-independent metrics.

5 Experiments

We run our experiments on Lambda Tensorbook laptop with Nvidia GeForce RTX 3080 16G RAM GPU. We set up the conda environment cs224n_dfp for model training and evaluation environment following the minBERT default final project instruction. We also installed additional python libraries (e.g., numpy, math) needed for out-of-domain detection implementation.

Data: CLINC dataset [1] is a benchmark NLU dataset that was collected in the same style made by real users of task-oriented systems. The queries cover 150 intents, plus out-of-scope queries that do not fall within any of the 150 in-scope intents. We use the CLINC plus version in Hugging Face datasets `https://huggingface.co/datasets/clinc_oos`. In our model training, we only use 15,000 train in-domain training data and select the model with the best accuracy on 3,000 validation data. We then test the in-domain performance on 4,500 in-domain test data and the out-of-domain detection performance on 1,000 out-of-domain test data. (Note that we do not use the 250 train and 100 validation out-of-domain data in either model training or testing.)

Evaluation Method: For in-domain classification evaluation, we use the *accuracy* metric. For out-of-domain detection evaluation, we use EER, FPR95, detection error, AUROC and AUPR IN/OUT metrics as described in section 4.

We evaluate on the models listed in Table 1. We consider two types of models, encoder models that only use Transformer encoders (a.k.a. representation model) and decoder models that only use Transformer decoders (a.k.a. generation model). For both encoder and decoder models, we evaluate in-domain classification and out-of-domain detection performance using their Sequence Classification model versions. For decoder models, we further use them to generate text and then leverage these generated texts to evaluate in-domain classification and out-of-domain detection performance.

Table 1: Summary of Models used in Our Experiment

Model Type	Model	Size	Pretraining data
Encoder model	Albert	46MB	BookCorpus, English Wikipedia (16GB of text)
	BertBase	419MB	BookCorpus, English Wikipedia (16GB of text)
	BertLarge	1.3GB	BookCorpus, English Wikipedia (16GB of text)
	RoBertaLarge	1.4GB	BookCorpus, English Wikipedia, CC-News, OpenWebText, Stories (160GB of text)
	MegatronBert	1.3GB	BookCorpus, English Wikipedia, CC-Stories, RealNews, OpenWebtext (174GB of deduplicated text)
Decoder model	GPT2Small	488MB	WebText (40GB of text)
	GPT2Medium	1.4GB	WebText (40GB of text)
	GPT3	175GB	Books, articles, websites, and other texts (570GB of text)

Experimental Details: We first processed CLINC dataset for the aforementioned different models: (1) Sequence Classification model in `prepare_clinc_huggingface.py` and (2) decoder generation model in `prepare_clinc_gpt.py` which generates the following format of training data:

"<sentence>. ask for <label text>"

where *<label text>* is the label name of each class in CLINC dataset. To use decoder generation model generated texts, we consider two ways: string matching with label text and feed augmented utterance (original utterance+generated text) into a classification model.

For all models, we finetune the whole model using CLINC dataset. We use learning rate 2e-5, batch size 64 to train each model 10 epochs. For each model, we run experiments 3 times then report the

mean and standard deviation of all evaluation metrics across the 3 models. For each model, we use the model checkpoint with the best in-domain classification accuracy on the validation dataset, to evaluate in-domain classification and out-of-domain detection.

For Monte-Carlo Dropout method, we only use MC dropout during model inference. We replace the dropout in all layers and for each model we run 3 stochastic passes by sampling the dropout masks. We then use the method in 4 on 3 stochastic models to detect OOD.

For GPT3 finetuning, we follow the instruction from OpenAI `https://platform.openai.com/docs/guides/fine-tuning`, using the prepared data described in experiment details in section 5. We use OpenAI default hyperparameters. After we finetuned GPT3, we wrote GPT3 inference in `gpt3_finetuning/gpt3_inference.py`. We generate text of maximum length 5 since the longest label text is 5.

6 Experimental Results & Analysis/Findings

Table 2: In-domain Classification Results

Model Type	Model	Training Data	In-domain Accuracy(↑) Validation	Test
Encoder Classifier	Albert	Original Training data	94.18±0.74	93.38±0.13
	BertBase		96.82±0.02	96.19±0.05
	BertLarge		97.00±0.05	96.49±0.13
	RoBertaLarge		98.19±0.16	97.09±0.22
	MegatronBert		97.97±0.05	97.32±0.04
	MegatronBert	GPT2 Medium Generated Text	57.79±0.44	59.37±0.02
	MegatronBert	GPT2 Medium Augmented Text	98.21±0.06	97.21±0.01
Decoder Classifier	GPT2Small	Original Training data	93.70±0.14	93.82±0.06
	GPT2Medium		96.80±0.05	96.16±0.08
Decoder Generation (String matching)	GPT2Medium	Original Training data	-	28.91±0.03
	GPT3		-	96.80±0.01

6.1 In-domain Classification Performance Analysis

Table 2 shows the in-domain classification accuracy of both encoder and decoder classifier (sequence classification) models. We have the following observations:

- Encoder classifier has slightly better in-domain accuracy than decoder classifier. This is because encoder models target on learning better representations of the whole sentence.
- The larger the model size is, the higher in-domain accuracy is (with marginal improvement). For similar size models, the model pretrained on larger corpus has better in domain accuracy than those pretrained on smaller corpus (e.g., MegatronBert > RoBertaLarge > BERTLarge).
- When using decoder GPT2 generated texts (both only generated text and augmented utterance) to train an additional encoder classifier MegatronBert, in-domain accuracy is worse than only using original CLINC training data. This is because oftentimes the generated text interpreted some other relevant meaning of the input sentence which confuses the additional classifier model. Below is an example:

  "how do germans say goodnight. ask for greeting"

  In this example, while greeting is relevant to the sentence, only using "greeting" as training data in the additional classifier will lose the translation semantics in the original sentence.
- Compared to similar size of classifier models, decoder generation model has worse accuracy when using string matching between generated text and label text. GPT3 finetuned model has much higher in-domain accuracy than GPT2 medium model.

6.2 Out-of-domain Detection Performance Analysis

For out-of-domain detection, we evaluated the following types of models.

Table 3: Encoder-based Classification Model Out-of-Domain Detection Results

Model	OOD Method	EER(↓)	FPR95(↓)	DET ERR(↓)	AUROC(↑)	AUPR IN(↑)	AUPR OUT(↑)
Albert	**Confidence**	11.50±0.14	27.74±0.74	11.27±0.28	94.47±0.41	98.52±0.20	80.86±0.10
	Entropy	10.60±0.28	22.98±1.32	10.49±0.24	95.04±0.40	98.67±0.18	82.77±0.47
	ODIN10	10.37±0.38	23.74±1.55	10.12±0.32	95.29±0.39	98.78±0.14	82.79±0.40
	ODIN100	10.40±0.42	23.76±1.44	10.12±0.31	95.31±0.38	98.79±0.13	82.82±0.40
	ODIN1000	10.40±0.42	23.76±1.43	10.13±0.31	95.31±0.38	98.79±0.13	82.79±0.40
	ODIN10000	10.40±0.42	23.75±1.43	10.13±0.31	95.31±0.38	98.79±0.13	82.37±0.42
	Mahalanobis	9.70±0.00	20.04±0.01	9.40±0.00	95.91±0.14	98.97±0.05	83.54±1.08
	MaxProbMC	26.67±2.50	69.95±4.21	26.21±2.17	80.45±2.66	94.20±0.83	47.33±4.16
	ODIN1000MC	29.53±2.17	79.44±1.62	29.24±2.30	76.73±2.74	93.19±1.05	39.55±2.34
	EntropyMC	30.50±3.54	79.61±3.84	30.10±3.49	75.81±3.95	92.90±1.17	39.29±4.56
BertBase	**Confidence**	8.77±0.19	16.22±0.49	8.75±0.22	96.48±0.11	99.07±0.06	87.38±0.40
	Entropy	8.57±0.33	13.69±0.39	8.46±0.26	96.87±0.13	99.16±0.06	89.15±0.20
	ODIN10	8.20±0.00	13.28±0.21	7.94±0.04	97.07±0.16	99.23±0.07	89.23±0.20
	ODIN100	8.17±0.09	13.28±0.12	7.86±0.01	97.07±0.17	99.22±0.07	89.26±0.17
	ODIN1000	8.17±0.09	13.28±0.11	7.86±0.00	97.07±0.17	99.22±0.07	89.23±0.17
	ODIN10000	8.17±0.09	13.28±0.11	7.85±0.01	97.07±0.17	99.22±0.08	88.86±0.17
	Mahalanobis	7.80±0.14	13.08±0.13	7.69±0.10	97.22±0.04	99.34±0.00	86.74±0.91
	MaxProbMC	9.13±0.09	16.76±0.04	8.83±0.11	96.60±0.06	99.11±0.03	87.71±0.27
	ODIN1000MC	8.27±0.05	14.19±0.66	7.82±0.22	97.07±0.10	99.26±0.03	88.64±0.09
	EntropyMC	8.53±0.09	13.83±0.26	8.09±0.07	97.07±0.11	99.25±0.04	89.46±0.04
BertLarge	**Confidence**	7.53±0.33	13.06±2.05	7.26±0.36	97.16±0.17	99.29±0.07	89.57±0.11
	Entropy	7.17±0.38	11.25±0.67	7.06±0.31	97.47±0.12	99.36±0.06	91.19±0.09
	ODIN10	6.77±0.24	9.90±0.06	6.64±0.25	97.67±0.12	99.42±0.07	91.37±0.06
	ODIN100	6.77±0.24	9.86±0.12	6.62±0.26	97.68±0.12	99.42±0.07	91.40±0.06
	ODIN1000	6.83±0.19	9.86±0.12	6.62±0.26	97.68±0.12	99.42±0.07	91.38±0.06
	ODIN10000	6.83±0.19	9.86±0.13	6.62±0.26	97.68±0.12	99.42±0.07	91.12±0.06
	Mahalanobis	6.30±0.00	8.74±0.54	6.15±0.05	97.90±0.02	99.50±0.00	91.55±0.29
	MaxProbMC	7.80±0.28	14.23±1.49	7.62±0.31	97.29±0.18	99.36±0.06	89.48±0.29
	ODIN1000MC	6.90±0.14	10.41±0.71	6.80±0.25	97.79±0.15	99.49±0.05	90.90±0.38
	EntropyMC	7.47±0.09	11.16±0.06	7.19±0.14	97.63±0.11	99.43±0.04	91.37±0.22
RoBertaLarge	**Confidence**	7.83±0.09	12.42±0.40	7.70±0.19	97.35±0.07	99.30±0.06	91.16±0.10
	Entropy	7.73±0.09	11.58±0.34	7.58±0.20	97.51±0.07	99.33±0.06	92.18±0.04
	ODIN10	7.40±0.28	10.86±0.56	7.27±0.25	97.66±0.06	99.37±0.05	92.56±0.10
	ODIN100	7.33±0.24	10.87±0.58	7.26±0.25	97.66±0.06	99.37±0.05	92.57±0.10
	ODIN1000	7.33±0.24	10.87±0.58	7.26±0.25	97.66±0.06	99.37±0.05	92.55±0.11
	ODIN10000	7.33±0.24	10.87±0.58	7.26±0.25	97.66±0.06	99.37±0.05	92.30±0.13
	Mahalanobis	6.37±0.05	8.32±0.02	6.05±0.03	98.39±0.02	99.60±0.01	94.31±0.00
	MaxProbMC	8.23±0.24	15.21±0.41	8.04±0.15	97.16±0.05	99.27±0.02	90.24±0.25
	ODIN1000MC	7.27±0.05	11.19±0.84	7.13±0.07	97.63±0.01	99.38±0.06	92.01±0.16
	EntropyMC	8.03±0.09	13.50±0.71	7.76±0.07	97.43±0.05	99.28±0.03	91.95±0.24
MegatronBert	**Confidence**	6.80±0.00	9.33±0.69	6.61±0.13	97.92±0.02	99.48±0.02	92.26±0.32
	Entropy	6.60±0.14	9.02±0.76	6.49±0.14	98.11±0.01	99.52±0.02	93.38±0.31
	ODIN10	6.27±0.24	8.31±0.58	6.14±0.16	98.19±0.02	99.54±0.03	93.53±0.22
	ODIN100	6.27±0.24	8.28±0.58	6.14±0.16	98.19±0.02	99.54±0.03	93.54±0.22
	ODIN1000	6.27±0.24	8.28±0.58	6.15±0.16	98.19±0.02	99.54±0.03	93.52±0.22
	ODIN10000	6.27±0.24	8.28±0.58	6.14±0.16	98.19±0.02	99.54±0.03	93.35±0.24
	Mahalanobis	5.90±0.14	7.06±0.33	5.75±0.08	98.38±0.05	99.61±0.01	93.42±0.42
	MaxProbMC	7.47±0.38	11.51±0.84	7.17±0.29	97.70±0.06	99.44±0.00	91.38±0.39
	ODIN1000MC	6.53±0.33	8.92±0.53	6.36±0.33	98.07±0.02	99.53±0.02	92.69±0.43
	EntropyMC	7.27±0.38	10.42±0.41	7.12±0.26	97.95±0.01	99.48±0.03	92.93±0.25

6.2.1 Encoder Classifier Models

Results are shown in Table 3. We summarize the following findings from Table 3.

- In general, larger models have better OOD detection performance. For models of similar size, the models pretrained on larger size of text corpus have better OOD detection performance.
- Among all OOD detection methods, model feature based OOD detection method Mahalanobis performs better than other OOD methods which are based on the last layer logits. This is because the model features are well trained during the model pretraining while the logits are mainly dependent on the new linear classifier layer that was not pretrained.

- Among all logits based OOD methods, ODIN performs best. Different temperature scales T only have little impact on the OOD detection performance, especially when T is large enough (e.g., $T > 100$).
- MC Dropout with MaxProb score function is less effective than the baseline confidence based method for Transformer models. When combining with ODIN, the performance improves but still worse than ODIN without MD dropout. For less capable models such as Albert, MC dropout leads to a significant OOD detection performance drop. This might be because we only use MC dropout during inference due to the limited time and resources.
- For small size models, better OOD detection methods (e.g., ODIN, Mahalanobis) will provide more OOD detection performance gain.

6.2.2 Decoder Classifier Models

Since the decoder classifier model does not have the sentence based feature, we only test on logits based OOD methods. As shown in Table 4, we find that most observations are similar to encoder classifier models.

Table 4: Decoder-based Classification Model Out-of-Domain Detection Results

Model	OOD Method	EER(↓)	FPR95(↓)	DET ERR(↓)	AUROC(↑)	AUPR IN(↑)	AUPR OUT(↑)
GPT2Small	**Confidence**	11.87±0.09	25.62±0.20	11.46±0.18	95.03±0.11	98.76±0.02	79.36±1.14
	Entropy	10.33±0.19	18.84±0.11	10.04±0.34	95.82±0.06	98.94±0.01	81.99±0.77
	ODIN10	9.83±0.47	19.88±0.29	9.67±0.43	95.93±0.08	98.95±0.08	81.88±0.74
	ODIN100	10.00±0.42	20.58±0.51	9.91±0.37	95.83±0.11	98.91±0.09	81.70±0.72
	ODIN1000	9.97±0.52	20.54±0.53	9.91±0.38	95.82±0.11	98.90±0.10	81.69±0.72
	ODIN10000	9.97±0.52	20.54±0.53	9.91±0.38	95.82±0.11	98.90±0.10	81.65±0.72
	MaxProbMC	11.77±0.24	24.33±0.44	11.46±0.12	95.14±0.14	98.72±0.05	81.99±1.58
	ODIN1000MC	10.57±0.09	20.78±0.14	10.39±0.05	95.59±0.03	98.86±0.08	82.60±1.29
	EntropyMC	10.27±0.09	19.36±0.16	10.12±0.11	95.77±0.16	98.91±0.02	83.65±1.73
GPT2Medium	**Confidence**	9.13±0.38	18.53±0.99	8.55±0.26	96.93±0.12	99.29±0.02	88.00±0.85
	Entropy	8.70±0.28	14.71±0.85	8.25±0.24	97.40±0.09	99.39±0.02	89.90±0.80
	ODIN10	7.30±0.14	10.22±0.71	7.06±0.12	97.63±0.14	99.40±0.08	90.96±0.36
	ODIN100	7.17±0.24	10.15±0.87	7.08±0.17	97.54±0.17	99.37±0.09	90.79±0.34
	ODIN1000	7.17±0.24	10.13±0.91	7.09±0.18	97.53±0.17	99.36±0.09	90.77±0.33
	ODIN10000	7.17±0.24	10.13±0.92	7.09±0.18	97.53±0.17	99.36±0.09	90.67±0.31
	MaxProbMC	9.57±0.19	18.52±1.54	9.03±0.06	96.71±0.08	99.22±0.01	87.96±0.54
	ODIN1000MC	7.97±0.09	12.00±0.36	7.68±0.18	97.28±0.18	99.31±0.08	90.09±0.01
	EntropyMC	8.30±0.00	13.17±0.68	8.17±0.00	97.31±0.02	99.35±0.00	89.93±0.45

6.2.3 Decoder Generation Model + String Matching

For GPT3 generated text, we consider it matches the label text if the label text is a substring of the generated text since GPT3 may generate additional words with similar semantic meanings which do not affect the classification accuracy. We also compute the averaged confidence (probability) of all generated words and study the trade-off between in-domain accuracy and out-of-domain false positive rate. Table 5 shows the best trade-off between them with confidence threshold 0.62. It is better than some weaker encoder classifier models but still worse than most encoder classifier models. For these encoder classifier models, we choose the out-of-domain threshold as the value which does not affect the in-domain classification accuracy.

Table 5: GPT3 Finetuning vs. Encoder-based Classification Models (with Mahalanobis)

Model	Training data (class label)	In-domain Accuracy	Out-of-domain FPR
GPT3	label text	95.1	39.8
Albert	label ID	93.4	59.7
BertBase		96.2	44.5
BertLarge		96.5	36.7
RoBertaLarge		97.1	26.2
MegatronBERT		97.3	25.6

Table 6: GPT2 Generation Model + MegatronBert Classifier Out-of-Domain Detection Results

Model&Data	OOD Method	EER(↓)	FPR95(↓)	DET ERR(↓)	AUROC(↑)	AUPR IN(↑)	AUPR OUT(↑)
GPT2 Medium Augmented Text + MegatronBert	**Confidence**	7.30±0.00	10.85±0.30	7.09±0.07	97.63±0.04	99.28±0.06	92.21±0.18
	Entropy	7.20±0.00	10.09±0.26	6.91±0.08	97.83±0.05	99.32±0.05	93.24±0.21
	ODIN10000	6.67±0.09	9.12±0.17	6.53±0.13	97.96±0.06	99.35±0.05	93.45±0.27
	Mahalanobis	6.20±0.14	8.39±0.47	5.87±0.03	98.32±0.06	99.57±0.02	94.31±0.21
	MaxProbMC	8.00±0.00	14.41±0.68	7.75±0.04	97.37±0.01	99.26±0.05	91.28±0.01
	ODIN1000MC	7.27±0.05	10.64±0.80	7.10±0.05	97.81±0.01	99.35±0.05	92.94±0.13
	EntropyMC	7.67±0.05	12.11±0.22	7.35±0.06	97.69±0.00	99.31±0.05	92.91±0.08

6.2.4 Decoder Generation Model Augmented Data + Encoder Classifier (MegatronBert)

We first implemented `data_preparation/prepare_clinc_aug.py` and prepared the augmented training data as follows: *"<sentence>. ask for <generated text>"*, where *<generated text>* is the text generated by GPT2 medium model finetuned on CLINC dataset. Then we train the best encoder classifier model MegatronBert using the augmented training data. As shown in Table 6, OOD detection performance is similar to just using original CLINC training data.

6.3 Qualitative Analysis

We conduct qualitative analysis based on the best model Megatron classifier coupled with the best OOD detection method Mahalanobis. We found two main categories of errors as shown in Table 7.

Ambiguous Sentences: Both in-domain and out-of-domain examples in the top part of Table 7 show a broader scope than the corresponding intents in CLINC dataset. For example, *can i fly with my razors or are there restrictions* could also talk about check-in luggages. *what's the way for delta to cancel a flight* is also relevant to the in-domain intent "flight_status".

Semantic Misunderstanding: The bottom part of Table 7 show the semantic misunderstanding of the sentences that lead to the errors. One type of misundertanding is the negation that deep learning models cannot correctly handle in many cases. In the example of *read my friend's text message*, the model captures the keyword "text" and confuses it with sending text messages. The last type of misunderstanding comes from the wrong tense understanding. The last row of Table 7 shows the tense misunderstanding leads to the confusion of "spending history" intent.

Table 7: Qualitative Analysis (MegatronBert + Mahalanobis)

	In-domain test data detected as out-of-domain	**Intent Ground Truth**	**Out-of-domain test data detected as in-domain**
Ambiguous Sentences	*can i fly with my razors or are there restrictions*	carry on	*what's the way for delta to cancel a flight*
	how do i ask the ai to help me with math problem	calculator	*how many prime numbers are there between 0 and 100*
Semantic Misunderstanding	*tell fred that i don't have his guitar*	text	*read my friend's text message*
	did i stick to my dinner budget	spending history	*what's my budget for today's shopping trip*

7 Future work

We plan to investigate more on MC dropout method for classification model by incorporating MC dropout during model training. We will also study the effectiveness of different uncertain estimation scores introduced in [13]. In addition, we plan to further explore OOD detection using decoder models like GPT. While encoder models seem to do better for OOD detection than decoder models, how to best use decoder only models for general natural language classification tasks is an interesting problem to study in near future. We also believe OOD problem is related to large language model's hallucination problem. While OOD detection problem is similar to a sentence level hallucination, there also exist token level hallucination problems in GPT models, leading to potential factual errors

from GPT models. As future work we wish to extend our work to detect the hallucination problem in large language models like GPT.

References

[1] 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 *EMNLP-IJCNLP*, pages 1311–1316, 2019.

[2] Shiyu Liang, Yixuan Li, and R Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *ICLR*, 2017.

[3] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *NeurIPS*, pages 7167–7177, 2018.

[4] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, volume 48, pages 1050–1059, 2016.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanaova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Association for Computational Linguistics (NAACL)*, 2018.

[6] Tom Brown, Benjamin Mann, Nick Ryder, and et.al. Language models are few-shot learner. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020.

[7] G. Tur and R. De Mori. *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*. Wiley, 2011.

[8] Bing Liu and Ian Lane. Attention-based recurrent neural network models for joint intent detection and slot filling. In *INTERSPEECH*, pages 685–689, 2016.

[9] Joo-Kyung Kim, Gokhan Tur, Asli Celikyilmaz, Bin Cao, and Ye-Yi Wang. Intent detection using semantically enriched word embeddings. In *SLT*, pages 414–419, 2016.

[10] Martin E Hellman. The nearest neighbor classification rule with a reject option. *IEEE Transactions on Systems Science and Cybernetics*, 6(3):179–185, 1970.

[11] Gokhan Tur, Anoop Deoras, and Dilek Hakkani-Tür. Detecting out-of-domain utterances addressed to a virtual personal assistant. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[12] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Association for Computational Linguistics (ACL)*, 2018.

[13] Artem Shelmanov, Evgenii Tsymbalov, Dmitri Puzyrev, Kirill Fedyanin, Alexander Panchenko, and Maxim Panov. How certain is your Transformer? In *EACL*, pages 1833–1840, Online, April 2021.

[14] Seonghan Ryu, Sangjun Koo, Hwanjo Yu, and Gary Geunbae Lee. Out-of-domain detection based on generative adversarial network. In *EMNLP*, pages 714–718, 2018.