A new approach for fine-tuning sentence transformers for intent classification and out-of-scope detection tasks

Tianyi Zhang[1,2,3]*, Atta Norouzian[3], Aanchan Mohan[1], Frederick Ducatelle[3]

[1]Northeastern University, Vancouver, BC, Canada,
[2]University of Victoria, Victoria, BC, Canada,
[3]Cerence Inc, Burlington, MA, USA,

tianyizhang2@uvic.ca , atta.norouzian@cerence.com, aa.mohan@northeastern.edu, frederick.ducatelle@cerence.com

Abstract

In virtual assistant (VA) systems it is important to reject or redirect user queries that fall outside the scope of the system. One of the most accurate approaches for out-of-scope (OOS) rejection is to combine it with the task of intent classification on in-scope queries, and to use methods based on the similarity of embeddings produced by transformer-based sentence encoders. Typically, such encoders are fine-tuned for the intent-classification task, using cross-entropy loss. Recent work has shown that while this produces suitable embeddings for the intent-classification task, it also tends to disperse in-scope embeddings over the full sentence embedding space. This causes the in-scope embeddings to potentially overlap with OOS embeddings, thereby making OOS rejection difficult. This is compounded when OOS data is unknown. To mitigate this issue our work proposes to regularize the cross-entropy loss with an in-scope embedding reconstruction loss learned using an auto-encoder. Our method achieves a 1-4% improvement in the area under the precision-recall curve for rejecting out-of-sample (OOS) instances, without compromising intent classification performance.

1 Introduction

Virtual assistant (VA) systems often can handle only a limited scope of intents. Out-of-scope (OOS) rejection refers to the ability of a VA to identify and reject incoming queries that are outside its scope. This is a difficult (Fang et al., 2023) and increasingly important task in many scenarios. Our work is inspired by VAs in cars, which nowadays often operate in a hybrid mode where processing of certain user requests is handled locally, while others are transmitted to the cloud for response retrieval. Responding to users' requests using on-device/embedded models is cost-effective, quick,

and, importantly, can safeguard sensitive information. Cloud models on the other hand are typically much bigger and can respond to a wider range of queries. In such a setting, it is important that the on-device natural language understanding (NLU) models not only identify user queries for intents that are in-scope but also accurately detect out-of-scope input so that they can be either routed to the cloud or ignored. Another important use case for OOS rejection is the combination of a light-weight, specialized VA that works tandem with large language models (LLMs) for free conversation with the user. Similar to the in-car use case, the specialized VA can be run *before* the LLM and capture a subset of the incoming queries. This increases cost-effectiveness and controllability of the full solution, provided that it has good OOS rejection capabilities.

The most common approach for intent classification while rejecting OOS samples is based on first generating an encoding for the sentences (Hendrycks et al., 2020; Podolskiy et al., 2021) and then performing classification on them. In both (Hendrycks et al., 2020) and (Podolskiy et al., 2021) it was shown that the most suitable sentence encoders for this purpose are transformer-based encoders. Based on the task's domain, one could use one of the several sentence encoders available in the HuggingFace sentence transformer library [1]. Fine-tuning sentence encoders on the domain-specific data leads to better intent classification accuracy. This fine-tuning typically is performed by applying a softmax to the sentence embeddings. At test time, the same softmax layer could be used to perform intent classification, however, the softmax tends to produce over-confident predictions even for OOS samples (Dhamija et al., 2018; Hendrycks and Gimpel, 2018). Hence, after fine-tuning, the softmax layer is removed from the model and other

---

*Work as a part of an internship at Cerence Inc.

[1]https://sbert.net/

classification approaches based on embedding similarities are used for intent classification and OOS rejection(Podolskiy et al., 2021).

This fine-tuning approach is shown to be effective in teaching the model the class-discriminative features (Fort et al., 2021) which in our task would result in a very good intent classification accuracy. However, fine-tuning without regularization could make the model forget some of the task-agnostic knowledge about general linguistic properties, which could help OOS detection (Chen et al., 2023). This shortcoming was tackled in (Zhou et al., 2021) by adding a regularization term based on contrastive loss. In this paper, we propose a new regularization term based on the global dispersion of in-scope sentence embeddings[2]. This is similar to the idea of deep one-class classification (Ruff et al., 2018), in which the model learns to project all in-scope samples into a relatively small neighborhood in the embedding space. In our approach, this is achieved by attaching an auxiliary autoencoder head to the fine-tuning architecture which reduces the global dispersion of the in-scpoe embeddings through minimization of reconstruction error. This approach is explained in detail in Section 3.

2 Related Work

There are largely two categories of approaches for detecting OOS samples when performing intent-classification. The first category is based on explicitly teaching the model to distinguish between in-scope and OOS samples by introducing OOS samples during training. This is done by adding an extra OOS class to the classifier (Larson et al., 2019; Qian et al., 2022; Choi et al., 2021; Zhan et al., 2021) or by adding an auxiliary loss function to the cross entropy loss to enforce the model to output a uniform probability distribution over in-scope classes when dealing with OOS samples (Zheng et al., 2020). These approaches only work if the OOS test samples are drawn from a distribution similar to that of the OOS training samples. In (Fang et al., 2023) the authors prove mathematically that it is not possible to detect samples outside of known distributions unless some conditions are met. This means for robust detection of OOS samples, the training OOS test samples have to represent a wide variety of possible distributions. While collecting such training samples is not feasi-

ble, synthesizing OOS samples using models like GANs (Ryu et al., 2018; Lee et al., 2018) and manifold learning (Goyal et al., 2020; Bhattacharya et al., 2023) have shown promise to make the decision boundary around in-system training samples as tight as possible.

The second category consists of approaches that rely only on in-scope training data without making any assumption about the OOS class. These approaches are largely based on sentence embeddings. Sentence embeddings generated by transformer encoders are shown to perform better than the ones generated using traditional NLU models (Hendrycks et al., 2020; Podolskiy et al., 2021). The classification of sentence embeddings into in-scope intent classes and into in-scope versus OOS could be done using non-parametric methods such as KNN (Zhou et al., 2022) or density based methods (Chen et al., 2023; Ren et al., 2021; Xu et al., 2020). There is a trade-off between the model footprint and its accuracy when it comes to choosing between parametric and non-parametric approaches. Due to constraints on the size of the model put in the car we chose the parametric approach based on the Mahalanobis distance.

The sentence embeddings could be generated using pretrained sentence transformers (Hendrycks et al., 2020) but fine-tuning the encoder for the task at hand provides more suitable embeddings (Darrin et al., 2024; Zhou et al., 2021; Barnabo et al., 2023; Zhou et al., 2022). The work in (Zhou et al., 2021) highlights that while fine-tuning based on cross-entropy loss effectively separates sentence embeddings of different intent classes, it struggles to differentiate between in-scope samples and OOS samples. In that paper, this issue is tackled by adding a secondary loss function to the fine-tuning based on contrastive loss. The contrastive loss increases the distance between intent classes in the embedding space while reducing the distance between embeddings of the same intent class. However, since this loss tries to push the in-scope intent classes as far as possible from each other, the intent classes could start overlapping with OOS samples in the embedding space. Our approach inspired by the one-class classification in (Ruff et al., 2018) tries to reduce the dispersion of the in-scope intent classes in the embedding space by replacing the contrastive loss with reconstruction loss obtained using an autoencoder.

---

3 Methodology

This section discusses the details of our modelling formalism. Sub-section 3.1 talks about our training cost-function(s), whereas sub-section 3.3 talks about our inference methodology.

3.1 Model Fine-tuning

Figure 1 shows our model architecture. Let $\mathbf{s}^i$ denote the d dimensional sentence embedding of the ith training sample generated after pooling the output of the transformer encoder. Here we use $\mathbf{y}^i$ to denote a C dimensional one-hot vector associating i^{th} input to one of C in-scope intents. The jth element of $\mathbf{y}_i$ namely y_j^i is equal to 1 if and only if $\mathbf{s}^i$ belongs to the jth class where $j \in \{1, \ldots, C\}$. In the baseline fine-tuning approach, a softmax layer is applied to $\mathbf{s}^i$ to map it to $\mathbf{e}^i$, a C-dimensional vector of probabilities. The cross-entropy loss $\mathcal{L}_{CE}^i$ of the ith training example is then calculated as:

$$\mathcal{L}_{CE}^i = -\sum_{j=1}^{C} y_j^i \log(e_j^i) \quad (1)$$

In the proposed fine-tuning approach, the sentence embedding $\mathbf{s}^i$ is passed to a second head which is comprised of an autoencoder network. The autoencoder reconstructs the embedding as $\mathbf{r}^i$. The reconstruction loss computed using mean-squared error is calculated as:

$$\mathcal{L}_{AE}^i = \frac{1}{d} \sum_{k=1}^{d} (s_k^i - r_k^i)^2 \quad (2)$$

The architecture of the model along with the size of the layers of the autoencoder head are provided in Section 4.1. The final loss is calculated as follows weighted sum of the two losses described above as

$$\mathcal{L}^i = (1 - \alpha)\mathcal{L}_{CE}^i + \alpha \mathcal{L}_{AE}^i \quad (3)$$

Here α tuned as a hyperparameter allows us to control the contribution of the individual losses towards the final loss.

3.2 Class-based Mean and Covariance Calculation

After training, the autoencoder and the softmax heads are discarded. The transformer encoder trained with Eq. (3) as the cost function is then primarily used for extracting sentence embeddings. Sentence embeddings using this transformer encoder are then generated for each training sample

belonging to one of the C in-scope intent classes. These per-class sentence embeddings are then used to construct a set of C mean-vectors $\boldsymbol{\mu}_j$ where $j \in 1, \ldots, C$. All of the training set sentence embeddings for the C classes are then used to calculate a universal covariance matrix $\boldsymbol{\Sigma}$.

3.3 Classification and Inference

For an incoming query q, if $\mathbf{s}^q$ is its corresponding sentence embedding, then the class-specific Mahalanobis distance d_j is calculated as follows:

$$d_j(\mathbf{s}^q) = \sqrt{(\mathbf{s}^q - \boldsymbol{\mu}_j)^\top \Sigma^{-1} (\mathbf{s}^q - \boldsymbol{\mu}_j)} \quad (4)$$

Once the distances are calculated, a minimum distance $d_{min}(\mathbf{s}^q)$ and the index $c_{min}(\mathbf{s}^q)$ of the candidate centroid is picked as follows.

$$d_{min}(\mathbf{s}^q) = \min_j d_j(\mathbf{s}^q) \quad (5)$$

$$c_{min}(\mathbf{s}^q) = \arg\min_j d_j(\mathbf{s}^q) \quad (6)$$

The quantity $d_{min}(\mathbf{s}^q)$ is then compared to a threshold τ to determine if the query q is in-scope or out-of-scope. This threshold is a hyper-parameter and is set empirically. If the query q is determined to be in-scope then $c_{min}(\mathbf{s}^q)$ is picked as the candidate class. This method of using a soft-max during training, but using the Mahalanobis distance during inference for classification is consistent with previous work (Podolskiy et al., 2021; Ren et al., 2021).

4 Experimental Setup

This section talks about our experimental setup. The main objectives of our experimental setup is to evaluate the capability of our proposed fine-tuning approach to improve the model's ability to detect OOS queries robustly while maintaining in-scope intents classification accuracy.

4.1 Sentence-encoder Configuration

The `bert-base-uncased` (Devlin et al., 2018) model followed by maxpooling was used to extract sentence embeddings. Sentence embeddings from the transformer sentence encoder have dimensionality $d = 768$. As shown in Figure 1 these embeddings pass through an autoencoder with a six-layer architecture designed to compress and reconstruct the sentence embeddings. The first 3 layers in the autoencoder reduce the data dimensionality from 768 to 512, 512 to 64, and finally
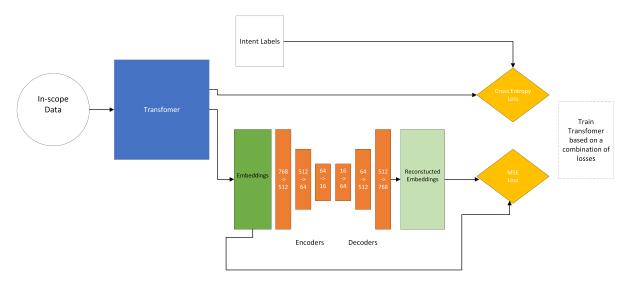
Figure 1: Model architecture for reducing the dispersion of in-scope embeddings. In-scope data is fine-tuned on a cross-entropy loss, with an auxiliary autoencoder loss.

from 64 to a 16-dimensional bottleneck. The subsequent 3 layers reconstruct the sentence embedding back to its dimensionality of $d = 768$. The transformer sentence-encoder is then trained using the objective function stated in Eq. (3). The errors are backpropagated from both heads back to the transformer sentence encoder. All layers of the transformer encoder model were fine-tuned.

4.2 Hyperparameter Optimization and Training

The training was found to be sensitive to the autoencoder weight parameter α. For this reason grid search was conducted for α with the following values $[0.01, 0.1, 0.2, 0.5, 0.9]$. The learning rate, batch size and no. of epochs were kept constant. It was found across the different validation sets that the optimal value for $\alpha = 0.1$. The performance started to deteriorate drastically for higher values of α.

The autoencoder weight α was then kept fixed for further hyperparameter optimization. Our work uses an open source hyperparameter optimization framework called Optuna (Akiba et al., 2019). Learning rates between 1×10^{-3} and 5×10^{-5} were explored using a logarithmic scale to prioritize smaller increments closer to the lower end of the spectrum, as transformer models often benefit from precise adjustments in learning rates. The number of training epochs ranged from 5 to 50. Batch size values were explored between 16, 32, 64, 128, 256, 512. The exact values for hyperparameters for each dataset appear in Appendix A.2.

4.3 Evaluation Metrics

The primary metric for assessing the effectiveness of our OOS detection was the Area Under the Precision-Recall curve (AUPR). It is important to mention that we label OOS samples as positive and in-scope samples as negative and hence we report AUPRood which signifies that. This metric is particularly suitable for comparing two binary classifiers when the test data is imbalanced like those with a high proportion of in-scope queries compared to OOS queries. The second metric is used is Area Under the ROC curve (AUROC). Our work additionally looks at the intent classification accuracy. This is important as our goal is to improve OOS rejection while maintaining in-scope intent classification accuracy.

4.4 Datasets

CLINC150 Dataset: The CLINC150 dataset (mis, 2020) is a benchmark dataset for evaluating natural language understanding systems particularly in the context of intent and slot filling tasks. The data set comprises 150 intent classes with an extra class labeled as out-of-scope. The training data consists of 15,000 examples with 100 examples per intent. The out-of-scope intent was not used in training. The validation data consists of 3,000 examples with 20 examples per intent. The test data consists of 4,500 examples with 30 examples per intent. The data spans across 10 diverse domains, such as banking, credit cards, kitchen appliances making it comprehensive for real-world scenarios. Each data sample consists of a short text utterance, paired with an

intent label.

Stackoverflow Dataset: This dataset is a curated subset from a challenge dataset originally published by Kaggle[3]. The selection includes question titles that have been categorized into 20 distinct intent classes following the methodology proposed by Xu et al. (Xu et al., 2017). Since this subset does not inherently include labeled out-of-scope (OOS) samples, we adopted the procedure described by Lin and Xu (Lin and Xu, 2019) to designate classes as either in-scope (IS) or OOS. Specifically, we retain classes that, combined, cover at least 75% of the total dataset as IS. The remaining classes are considered OOS, and their instances are removed from the training dataset but retained and relabeled as OOS in the validation and test datasets. The specific details of dataset construction is detailed in Appendix A.1.1

MTOP Dataset: The MTOP dataset is a task-oriented dialogue dataset with a hierarchical structure of intent labels. In our experiments, we focus solely on the root-label of these intents. We utilized the English portion of this dataset, referred to as MTOP-EN, which comprises 87 intent classes in 11 domains. This dataset does not include a pre-defined out-of-scope (OOS) class. Based on the amount of data, the 'timer' domain is chosen as the pre-defined OOS class. Our preprocessing filtered out in-scope (IS) domains with fewer than 10 occurrences per IS class. The in-scope data was then split into training, validation, and testing sets using a stratified approach based on intent labels to maintain an equal distribution of intents across these splits. We allocate OOS data between validation and testing sets, without stratification, due to the uniform label of OOS.

Car Assistant Dataset: This is an internal dataset. Due to it's original massive size, we randomly selected around 200,000 utterances used per run for training, validation and testing. This in-scope part of the dataset is derived from user interactions with car assistant systems and contains 46 distintc intent classes while. The OOS part is constructed from 14 different setes including sms messages, dictated emails, book snippets, tweets, internet-scraped text and some other unsupported text phrases.

---

5 Results and Discussion

The OOS detection performance and intent classification accuracy of both the baseline and the proposed fine-tuning approaches are presented in Table 1. The table has 4 rows and 7 columns. Each row of Table 1 contains results on one particular dataset. The first three columns show dataset name and a summary of numbers of utterances in each dataset. The fourth column shows the fine-tuning cost function used namely cross-entropy (CE), versus the joint cross-entropy and autoencoder (CE+AE) fine-tuning objective introduced in Eq. (3). The next three columns display our results for the evaluation metrics mentioned in Section 4.3. As mentioned in the table caption, AUPRoos refers to calculating the AUPR by treating the OOS class in the test set as the positive class. AUROC refers the area under the receiver operating curve, and accuracy refers to intent classification accuracy using Eqs. (5) and (6). The intent classification accuracy is expressed as a percentage.

The results show that in 3 out of 4 datasets we tested, the proposed method improved OOS detection, while maintaining the same in-scope intent classification accuracy. Specifically, the relative improvement with regard to AUPRoos is seen to be 3.22% on the StackOverflow dataset, 3.45% on the MTOP dataset and 1.15% on the Car Assistant dataset. Due to its larger test set, the improvement on our internal car assistant dataset is statistically more significant than the improvement on the other two test sets. This can be attributed to the presence of a larger training set, which enables the autoencoder head to exert greater influence over the more than 100 million parameters of the sentence encoder.

5.1 Embedding Dispersion

We also measured the dispersion of the sentence embedding vector after baseline fine-tuning and after our proposed fine-tuning as shown in Table 2. The dispersion was calculated as follows. For each training dataset that appears in Table 2, training sentence embeddings were extracted first using our baseline cross-entropy (CE) model, and further using our model trained with joint cross-entropy and autoencoder objective (CE+AE). After extracting embeddings a global covariance matrix was calculated in each case namely Σ_{CE} and Σ_{CE+AE}. To measure dispersion, the trace of each of these matrices were calculated (Johnson et al., 2002). The

Dataset	#Train	#Test(is/oos)	Fine-tuning	AUPRoos	AUROC	Intent Classification Accuracy (%)
CLINC150	15,000	4,500 / 1,000	CE	0.916 ± 0.007	0.977 ± 0.001	95.8
			CE+AE	0.918 ± 0.004	0.978 ± 0.004	95.8
StackOverflow	79,048	16,940 / 14,617	CE	0.822 ± 0.053	0.881 ± 0.028	91.2
			CE+AE	**0.849 ± 0.050**	**0.893 ± 0.030**	90.9
MTOP	14,465	4,134 / 997	CE	0.869 ± 0.018	0.974 ± 0.004	97.0
			CE+AE	**0.899 ± 0.039**	**0.979 ± 0.009**	97.0
Car Assistant	600k	150k / 200k	CE	0.954 ± 0.005	0.959 ± 0.002	96.5
			CE+AE	**0.965 ± 0.004**	**0.966 ± 0.003**	96.6

Table 1: Comparison of cross-entropy (CE) fine-tuning and versus the joint cross-entropy and autoencoder objective (CE+AE). Here AUPRoos refers to the AUPR metric treating the OOS class as the positive class in the test set. The last column shows the intent classification accuracy result as a percentage.

Dataset	CE	CE+AE
CLINC150	17.767	17.762
StackOverflow	16.854	16.026
MTOP	17.269	16.744

Table 2: Dispersion of fine-tuned models

dispersion values thus calculated appear in Table 2. The dispersion values illustrate that global dispersion of in-scope embeddings is smaller when our proposed fine-tuning is applied. It can be observed that the smaller the dispersion gets the higher OOS detection accuracy becomes when comparing the two fine-tuning approaches. This supports our argument that constraining the in-scope embeddings in a smaller neighborhood in the embedding space helps in the separation of in-scope and OOS samples.

5.2 Replacing the Model with a Large Language Model (LLM)

Given the undeniable power of LLMs, one would naturally wonder what if the classification pipeline based on sentence encoder was replaced by an LLM. In other words, how well would a LLM perform intent classification and OOS detection tasks without fine-tuning and just by prompt engineering. To answer this question we examined the performance of ChatGPT's gpt-3.5-turbo-0125 model from OpenAI on the MTOP dataset. We evaluated the performance of the LLM for intent-classification and OOS detection separately with different prompts as we noticed that if we ask the LLM to do both tasks, it will overwhelmingly classify most samples as OOS. Furthermore, due to the limitations in context size, we were limited to use

200 training examples but we made sure that there is at least one sample for each intent in the training set. In our setup, the system prompt is followed by the user prompt in which the model is provided with training sentences and a single test sentence. The exact system prompt is included in the Appendix A.3. Each experiment was repeated five times and the mean values of AUPR and AUROC as well as classification precision are presented in Table 3.

Metric	Value
Average AUPR	0.624 ± 0.00440
AUROC	0.642
Intent Classification Acc.(%)	82.9

Table 3: Benchmark results on GPT-3.5

It is worth noting that even with a very limited amount of training data the LLM does a good job of classifying 82.9% of the samples correctly. However, detecting OOS samples just by looking at a few in-scope samples is proven to be a more difficult task even for the LLM. Although comparing the performance of our approach to the LLM performance for this task is not fair because the latter only saw a fraction of the training samples, it shows that one could not simply replace the classifier with an LLM and expect high intent classification and OOS detection accuracy.

Conclusion

In this paper, we introduce a new approach to fine-tuning sentence transformers used for intent classification, to improve their ability to detect OOS samples. We showed that sentence embeddings generated from encoders fine-tuned using the pro-

posed approach provide better separation between in-scope and OOS samples while maintaining the separation between intent classes.

Limitations

A limitation of our approach is that it requires more than a few examples per intent class during fine-tuning to make a big enough impact on the sentence encoder to improve OOS detection. In other words, it is not suitable for few-shot learning. This can be seen in the results given in Table 1 where the OOS accuracy stays the same for the CLINC150 dataset, where the ratio of samples to intent classes is much smaller than for the other datasets. The proposed approach was not evaluated for compositional or compound queries that contain both in-scope and OOS elements. This was mainly because in most virtual assistant systems the multi-intent queries are first broken into single-intent phrases, and then the classification step is performed. In addition, there are not many studies in the literature on this use case and not having publicly available datasets with such queries in them would make it difficult for us to benchmark our approach against SOTA approaches.

Acknowledgments

References

2020. CLINC150. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5MP58.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Giorgio Barnabo, Antonio Uva, Sandro Pollastrini, Chiara Rubagotti, and Davide Bernardi. 2023. Supervised clustering loss for clustering-friendly sentence embeddings: An application to intent clustering. In *IJCNLP-AACL 2023*.

Arindam Bhattacharya, Ankit Gandhi, Vijay Huddar, Ankith M S, Aayush Moroney, Atul Saroop, and Rahul Bhagat. 2023. Beyond hard negatives in product search: Semantic matching using one-class classification (smocc). In *WSDM 2023*.

Sishuo Chen, Wenkai Yang, Xiaohan Bi, and Xu Sun. 2023. Fine-tuning deteriorates general textual out-of-distribution detection by distorting task-agnostic features. In *EACL 2023*, page 564–579.

DongHyun Choi, Myeong Cheol Shin, EungGyun Kim, and Dong Ryeol Shin. 2021. Outflip: Generating out-of-domain samples for unknown intent detection with natural language attack. *arXiv preprint arXiv:2105.05601*.

Maxime Darrin, Guillaume Staerman, Eduardo Dadalto Câmara Gomes, Jackie CK Cheung, Pablo Piantanida, and Pierre Colombo. 2024. Unsupervised layer-wise score aggregation for textual ood detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17880–17888.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Akshay Raj Dhamija, Manuel Günther, and Terrance E. Boult. 2018. Reducing network agnostophobia. *ArXiv*, abs/1811.04110.

Zhen Fang, Yixuan Li, Jie Lu, Jiahua Dong, Bo Han, and Feng Liu. 2023. Is out-of-distribution detection learnable? *Preprint*, arXiv:2210.14707.

Stanislav Fort, Jie Ren, and Balaji Lakshminarayanan. 2021. Exploring the limits of out-of-distribution detection. In *Advances in Neural Information Processing Systems*, volume 34, pages 7068–7081. Curran Associates, Inc.

Sachin Goyal, Aditi Raghunathan, Moksh Jain, Harsha Vardhan Simhadri, and Prateek Jain. 2020. Drocc: Deep robust one-class classification. *Preprint*, arXiv:2002.12718.

Dan Hendrycks and Kevin Gimpel. 2018. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *Preprint*, arXiv:1610.02136.

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

Richard Arnold Johnson, Dean W Wichern, et al. 2002. Applied multivariate statistical analysis.

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. 2019. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1311–1316, Hong Kong, China. Association for Computational Linguistics.

Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. 2018. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *Preprint*, arXiv:1711.09325.

Ting-En Lin and Hua Xu. 2019. Deep unknown intent detection with margin loss. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5491–5496, Florence, Italy. Association for Computational Linguistics.

Alexander Podolskiy, Dmitry Lipin, Andrey Bout, Ekaterina Artemova, and Irina Piontkovskaya. 2021. Revisiting mahalanobis distance for transformer-based out-of-domain detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13675–13682.

Cheng Qian, Haode Qi, Gengyu Wang, Ladislav Kunc, and Saloni Potdar. 2022. Distinguish sense from nonsense: Out-of-scope detection for virtual assistants. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 502–511, Abu Dhabi, UAE. Association for Computational Linguistics.

Jie Ren, Stanislav Fort, Jeremiah Liu, Abhijit Guha Roy, Shreyas Padhy, and Balaji Lakshminarayanan. 2021. A simple fix to mahalanobis distance for improving near-ood detection. *Preprint*, arXiv:2106.09022.

Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. 2018. Deep one-class classification. In *International conference on machine learning*, pages 4393–4402. PMLR.

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, Brussels, Belgium. Association for Computational Linguistics.

Hong Xu, Keqing He, Yuanmeng Yan, Sihong Liu, Zijun Liu, and Weiran Xu. 2020. A deep generative distance-based classifier for out-of-domain detection with mahalanobis space. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1452–1460. International Committee on Computational Linguistics.

Jiaming Xu, Bo Xu, Peng Wang, Suncong Zheng, Guanhua Tian, and Jun Zhao. 2017. Self-taught convolutional neural networks for short text clustering. *Neural Networks*, 88:22–31.

Li-Ming Zhan, Haowen Liang, Bo Liu, Lu Fan, Xiao-Ming Wu, and Albert Lam. 2021. Out-of-scope intent detection with self-supervision and discriminative training. *arXiv preprint arXiv:2106.08616*.

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.

Wenxuan Zhou, Fangyu Liu, and Muhao Chen. 2021. Contrastive out-of-distribution detection for pretrained transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1100–1111.

Yunhua Zhou, Peiju Liu, and Xipeng Qiu. 2022. Knn-contrastive learning for out-of-domain intent classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5129–5141.

A Appendix

A.1 Details of dataset construction

A.1.1 Stackoverflow dataset

The dataset is divided into training, validation, and testing sets using a stratified split approach. The stratification ensures that the relative frequency of IS and OOS labels is maintained across the splits. This procedure is replicated across five different IS-OOS class configurations (splits), each initiated with a unique random seed for repeatability. For each split, the dataset undergoes:

1. Filtering to include only the specific 20 categories from the original dataset. The labels selected for inclusion in this subset are as follows: 'svn', 'oracle', 'bash', 'apache', 'excel', 'matlab', 'cocoa', 'visual-studio', 'osx', 'wordpress', 'spring', 'hibernate', 'scala', 'sharepoint', 'ajax', 'drupal', 'qt', 'haskell', 'linq', 'magento'.

2. Random shuffling and selection of tags to meet the 75% threshold for IS designation.

3. Out-of-domain data, not meeting the IS criteria, is split equally into validation and test sets, labeled as OOS.

A.2 Exact hyper-parameter values obtained for various datasets

Dataset	α	lr	bs	# epochs
CLINC150	0.1	10^{-4}	256	15
StackOverflow	0.1	5×10^{-5}	1024	6
MTOP	0.1	2.25×10^{-5}	128	10
Car Assistant	0.1	2.25×10^{-5}	1024	7

Table 4: Parameter values for different datasets. Here α refers to the autoencoder importance, lr refers to the learning rate, bs refers to the batch size and # epochs refers to the number of epochs.

A.3 Prompt given to ChatGPT 3.5

In order to evaluate our approach against ChatGPT 3.5 we used the following system prompt for OOS detection task:

```
You are an AI assistant
    specialized in
    natural language
    processing tasks. You
    will be provided
    with training samples
    consisting of
    sentences and their
    corresponding intents
    . Your task is to
    determine whether a
    given sentence is in-
    scope (belongs to a
    known intent) or out-
    of-scope (does not
    belong to any known
    intent). Based on the
    provided training
    data, classify each
    input sentence and
    return a JSON object
    indicating whether
    the sentence is in-
    scope or out-of-scope
    . If the sentence is
    in-scope, also
    provide the intent
    name. If the sentence
    is out-of-scope,
    indicate that it is
    out-of-scope. The in-
    scope intents must
    match exactly with
```

```
    the intents provided
    in the training data
    except for oos.
    Instructions: 1. For
    each input sentence,
    determine if it is in
    -scope or out-of-
    scope based on the
    provided training
    data. 2. If the
    sentence is in-scope,
    return a JSON object
    with { inscope: true
    , scope: "intent_name
    " }. The intent name
    must match exactly
    with the intents
    provided in the
    training data. 3. If
    the sentence is out-
    of-scope, return a
    JSON object with {
    inscope: false, scope
    : "oos" }.
```

System Prompt for classification task:

```
You are an AI assistant
    specialized in
    natural language
    processing tasks. You
    will be provided
    with training samples
    consisting of
    sentences and their
    corresponding intents
    . Your task is to
    classify a given
    sentence's intent.
    Based on the provided
    training data,
    classify the input
    sentence and return a
    JSON object
    indicating the intent
    of the sentence. The
    intents must match
    exactly with the
    intents provided in
    the training data.
    Return a JSON object
    with { intent: "
```

```
    intent_name" }.
```