Abstract

Out-of-Domain (OOD) intent detection is important for practical dialog systems. To alleviate the issue of lacking OOD training samples, some works propose synthesizing pseudo OOD samples and directly assigning one-hot OOD labels to these pseudo samples. However, these one-hot labels introduce noises to the training process because some “hard” pseudo OOD samples may coincide with In-Domain (IND) intents. In this paper, we propose an adaptive soft pseudo labeling (ASoul) method that can estimate soft labels for pseudo OOD samples when training OOD detectors. Semantic connections between pseudo OOD samples and IND intents are captured using an embedding graph. A co-training framework is further introduced to produce resulting soft labels following the smoothness assumption, i.e., close samples are likely to have similar labels. Extensive experiments on three benchmark datasets show that ASoul consistently improves the OOD detection performance and outperforms various competitive baselines.

1 Introduction

Intent detection is essential for dialogue systems, and current methods usually achieve high performance under the closed-world assumption (Shu et al., 2017), i.e., data distributions are static, and only a fixed set of intents are considered. However, such an assumption may not be valid in practice, where we usually confront an open-world (Fei and Liu, 2016), i.e., unknown intents that are not trained may emerge. It is necessary to equip dialogue systems with Out-of-Domain (OOD) detection abilities so that they can accurately classify known In-Domain (IND) intents while rejecting unknown OOD intents (Yan et al., 2020a; Shen et al., 2021).

A major challenge for OOD detection is the lack of OOD samples (Xu et al., 2020a). In most applications, it is hard, if not impossible, to collect OOD samples from the test distribution before training (Du et al., 2021). To tackle this issue, various studies try to synthesize pseudo OOD samples in the training process. Existing methods include distorting IND samples (Choi et al., 2021; Shu et al., 2021; Ouyang et al., 2021), using generative models (Ryu et al., 2018; Zheng et al., 2020a), or even mixing-up IND features (Zhou et al., 2021a; Zhan et al., 2021). Promising results are reported by training a \((k + 1)\)-way classifier \((k\) IND classes + 1 OOD class) using these pseudo OOD samples (Geng et al., 2020). This classifier can classify IND intents while detecting OOD intent since inputs that fall into the OOD class are regarded as OOD inputs.

Previous studies directly assign one-hot OOD labels to pseudo OOD samples when training the \((k + 1)\)-way classifier (Shu et al., 2021; Chen and Yu, 2021). However, this scheme brings noise to the training process because “hard” pseudo OOD samples, i.e., OOD samples that are close to IND distributions, may carry IND intents (Zhan et al., 2021) (See Figure 1). Indiscriminately assigning one-hot OOD labels ignores the semantic connections between pseudo OOD samples and IND intents. Moreover, this issue becomes more severe...
as most recent studies are dedicated to producing hard pseudo OOD samples (Zheng et al., 2020a; Zhan et al., 2021) since these samples are reported to facilitate OOD detectors better (Lee et al., 2017). Collisions between pseudo OOD samples and IND intents will be more common.

We argue that ideal labels for pseudo OOD samples should be soft labels that allocate non-zero probabilities to all intents (Hinton et al., 2015; Müller et al., 2019). Specifically, we demonstrate in §3.2 that pseudo OOD samples generated by most existing approaches should be viewed as unlabeled data since they may carry both IND and OOD intents. Soft labels help capture the semantic connections between pseudo OOD samples and IND intents. Moreover, using soft labels also conforms to the smoothness assumption, i.e., samples close to each other are likely to receive similar labels. This assumption lays a foundation for modeling unlabeled data in various previous works (Luo et al., 2018; Van Engelen and Hoos, 2020).

In this study, we propose an adaptive soft pseudo labeling (ASoul) method that can estimate soft labels for given pseudo OOD samples and thus help to build better OOD detectors. Specifically, we first construct an embedding graph using supervised contrastive learning to capture semantic connections between pseudo OOD samples and IND intents. Following the smoothness assumption, a graph-smoothed label is produced for each pseudo OOD sample by aggregating nearby nodes on the graph. A co-training framework with two separate classification heads is introduced to refine these graph-smoothed labels. Concretely, the prediction of one head is interpolated with the graph-smoothed label to produce the soft label used to enhance its peer head. The final OOD detector is formulated as a \((k+1)\)-way classifier with adaptive decision boundaries.

Extensive experiments on three benchmark datasets demonstrate that ASoul can be used with a wide range of OOD sample generation approaches and consistently improves the OOD detection performance. ASoul also helps achieve new State-of-the-art (SOTA) results on benchmark datasets. Our major contributions are summarized:

1. We propose ASoul, a method that can estimate soft labels for given pseudo OOD samples. ASoul conforms to the important smoothness assumption for modeling unlabeled data by assigning similar labels to close samples.

2. We construct an embedding graph to help capture the semantic connections between pseudo OOD samples and IND intents. A co-training framework is further introduced to produce the resulting soft labels with the help of two separate classification heads.

3. We conduct extensive experiments on three benchmark datasets. The results show that ASoul consistently improves the OOD detection performance, and it obtains new SOTA results.

2 Related Work

OOD Detection: OOD detection problems have been widely investigated in conventional machine learning studies (Geng et al., 2020). Recent neural-based methods try to improve the OOD detection performance by learning more robust representations on IND data (Zhou et al., 2021c, 2022; Yan et al., 2020b; Zeng et al., 2021b). These representations can be used to develop density-based or distance-based OOD detectors (Lee et al., 2018b; Podolskiy et al., 2021; Liu et al., 2020; Tan et al., 2019). Some methods also propose to distinguish OOD inputs using thresholds based methods (Gal and Ghahramani, 2016; Lakshminarayanan et al., 2017; Ren et al., 2019; Gangal et al., 2020; Ryu et al., 2017), or utilizing unlabeled IND data (Xu et al., 2021; Jin et al., 2022).

Pseudo OOD Sample Generation: Some works try to tackle OOD detection problems by generating pseudo OOD samples. Generally, four categories of approaches are proposed: 1. Phrase Distortion (Chen and Yu, 2021): OOD samples are generated by replacing phrases in IND samples; 2. Feature Mixup (Zhan et al., 2021): OOD features are directly produced by mixing up IND features (Zhang et al., 2018); 3. Latent Generation (Marek et al., 2021): OOD samples are drawn from the low-density area of a latent space; 4. Open-domain Sampling (Hendrycks et al., 2018): data from other corpora are directly used as pseudo OOD samples. With these pseudo OOD samples, the OOD detection task can be formalized into a \((k+1)\)-way classification problem \((k\) is the number of IND intents). Our method can be combined with all the above OOD generation approaches to improve the OOD detection performance.

Soft Labeling: Estimating soft labels for inputs has been applied in a wide range of studies such as knowledge distillation (Hinton et al., 2015; Gou et al., 2021; Zhang et al., 2020), confidence cal-
ibration (Müller et al., 2019; Wang et al., 2021), or domain shift (Ng et al., 2020). However, few studies try to utilize this approach in OOD detection methods. Existing approaches only attempt to assign dynamic weights (Ouyang et al., 2021) or soft labels to IND samples (Cheng et al., 2022). Our method ASoul is the first attempt to estimate soft labels for pseudo OOD samples.

**Semi-Supervised Learning:** Our work is also related to semi-supervised learning (SSL) since they all attempt to utilize unlabeled data and share the same underlying smoothness assumption (Wang and Zhou, 2017; Lee et al., 2013; Li et al., 2021). Moreover, the co-training framework in ASoul also helps to enforce the low-density assumption (a variant of the smoothness assumption) (Van Engelen and Hoos, 2020; Chen et al., 2022) by exploring low-density regions between classes.

## 3 Background

### 3.1 Problem Definition

OOD intent detectors aim to classify IND intents while detecting OOD inputs. Concretely, given $k$ known IND intent classes $I = \{I_i\}_{i=1}^k$, the training set $D_I = \{(x_i, y_i)\}$ only contains IND samples, i.e., $x_i$ is an input, and $y_i \in I$ is the label of $x_i$. The test set $D = \{(\tilde{x}_i, \tilde{y}_i)\}$ consists both IND and OOD samples, i.e., $\tilde{y}_i \in I \cup \{I_{k+1}\}$, in which $I_{k+1}$ is a special OOD intent class. For a testing input $\tilde{x}_i$, an OOD detector should classify the intent of $\tilde{x}_i$ if $\tilde{x}_i$ belongs to an IND intent or reject $\tilde{x}_i$ if $\tilde{x}_i$ belongs to the OOD intent.

### 3.2 Analyzing Pseudo OOD Samples

Recent works have demonstrated that “hard” OOD samples, i.e., OOD samples akin to IND distributions, are more efficient in improving the OOD detection performance (Lee et al., 2018a; Zheng et al., 2020a). Promising performances are obtained using these hard samples on various benchmarks (Zhan et al., 2021; Shu et al., 2021).

However, we notice that hard pseudo OOD samples used in previous approaches may coincide with IND samples and carry IND intents. Besides Figure 1, we further demonstrate this issue by visualizing pseudo OOD samples produced by Zhan et al. (2021). Specifically, pseudo OOD samples are synthesized using convex combinations of IND features. Figure 2 shows the results on the Banking dataset (Casanueva et al., 2020) when 25% intents are randomly selected as IND intents. It can be seen that some pseudo OOD samples fall into the cluster of IND intents, and thus it is improper to assign one-hot OOD labels to these samples.

The above issue is also observed in other pseudo OOD sample generation approaches. Specifically, we implement the phrase distortion approach proposed by Shu et al. (2021) and employ crowd-sourced workers to annotate 1,000 generated pseudo OOD samples. Results show that up to 39% annotated samples carry IND intents (see Appendix A for more examples).
4 Method

4.1 Overview

In this study, we build the OOD intent detector following three steps: 1. Construct a set of pseudo OOD samples \( \mathcal{D}_P \); 2. Estimate a soft label for each sample \( x \in \mathcal{D}_P \); 3. Obtain a \((k+1)\)-way classifier and learn a decision boundary for each class to build an OOD detector. A testing input \( x \) is identified as OOD if \( x \) belongs to the OOD intent \( I_{k+1} \) or \( x \) is out of all decision boundaries.

Before applying ASoul, we assume a set of pseudo OOD samples \( \mathcal{D}_P \) are already generated using existing approaches. Figure 3 shows an overview of ASoul. Specifically, a shared utterance encoder \( f \) encodes each input \( x \in \mathcal{D}_I \cup \mathcal{D}_P \) into a representation, and an embedding projection head \( h \) constructs an embedding graph on these representations. A co-training framework is also implemented using two \((k+1)\)-way classification heads \( g_1 \) and \( g_2 \), and the prediction of one head is used to enhance soft labels of the peer head.

Note that ASoul is independent of specific methods to produce pseudo OOD samples in \( \mathcal{D}_P \). In this study, we test various approaches to obtain \( \mathcal{D}_P \).

4.2 Embedding Graph

Embedding Space: An embedding space is maintained in ASoul to capture semantic of input samples. Specifically, for an input \( x_i \), an encoder \( f \) is used to convert \( x_i \) into a representation vector, then a projection head \( h \) is used to map \( f(x_i) \) into an L2 normalized embedding \( z_i = h[f(x_i)] \) to construct the embedding space. To capture better semantic representations, a supervised contrastive loss (Khosla et al., 2020; Gunel et al., 2020) \( \mathcal{L}_{\text{ctr}} \) is optimized on labeled IND samples in \( \mathcal{D}_I \):

\[
\mathcal{L}_{\text{ctr}} = \sum_{x_i \in \mathcal{D}_I} -\frac{1}{|S(i)|} \sum_{x_j \in S(i)} \log \frac{e^{\Phi(x_i) \cdot \Phi(x_j)/t}}{\sum_{x_k \in A(i)} e^{\Phi(x_i) \cdot \Phi(x_k)/t}},
\]

where \( S(i) \) represents samples that share the same label with \( x_i \) in the current batch, \( A(i) \) represents all samples in the current batch except \( x_i \), \( \Phi \) maps an input \( x \) to its corresponding embedding (i.e., \( \Phi(x) = h[f(x)] \)), and \( t > 0 \) is a scalar that controls the separation of classes. \( \mathcal{L}_{\text{ctr}} \) captures the similarities between examples in the same class and contrast them with the examples from different classes (Gunel et al., 2020).

Graph-Smoothed Label: After obtaining the embedding space, we construct a fully connected unidirectional embedding graph \( \mathcal{G} \) using samples in \( \mathcal{D}_{IP} = \mathcal{D}_I \cup \mathcal{D}_P \). Specifically, we first map each sample \( x \in \mathcal{D}_{IP} \) into an embedding \( z \), i.e., \( z = \Phi(x) \), and then use all these embeddings as nodes for \( \mathcal{G} \). Every two nodes \( z_i \) and \( z_j \) in \( \mathcal{G} \) are linked with an edge. Moreover, we also assign a prior label \( l_p(x) \in \mathbb{R}^{k+1} \) to each sample \( x \in \mathcal{D}_{IP} \) to represent its annotation, i.e., for an IND sample \( x \in \mathcal{D}_I \), \( l_p(x) \) is defined as the one-hot label corresponding to \( y \), and for a pseudo OOD sample \( x \in \mathcal{D}_P \), \( l_p(x) \) is defined as the one-hot OOD label corresponding to \( I_{k+1} \).

For each OOD sample \( x \in \mathcal{D}_P \), a graph-smoothed label \( l_g(x) \) is obtained by aggregating adjacent nodes on \( \mathcal{G} \). Specifically, to conform to the smoothness assumption, we try to minimize the following distance when determining \( l_g(x) \):

\[
a_j = \frac{\exp(z \cdot z_j/\tau)}{\sum_{k=1}^{D_{IP}} \exp(z \cdot z_k/\tau)}.
\]

where \( 0 \leq \alpha \leq 1 \) is a scalar, \( d \) is a distance function, \( \tau > 0 \) is a scalar temperature. The second term in Eq. 2 enforces the smoothness assumption by encouraging \( l_g(x) \) to have similar labels with its nearby samples, whereas the first term tries to maintain \( l_g(x) \) to meet its original annotation \( l_p(x) \). For simplicity, we implement \( d \) as the Euclidean distance here, and thus minimizing Eq. 2 yields:

\[
l_g(x) = \alpha \cdot l_p(x) + (1 - \alpha) \sum_{x_j \in \mathcal{D}_{IP}} a_j \cdot l_p(x_j)
\]

Note that the result we derived in Eq. 3 follows most previous graph-smoothing approaches in semi-supervised learning (Van Engelen and Hoos, 2020). To the best of our knowledge, we are the first to apply this scheme to OOD detection tasks.

4.3 Co-Training Framework

To further enforce the smoothness assumption, a co-training framework is introduced in ASoul to learn better soft labels using \( l_g(x) \). Specifically, we implement two classification heads \( g_1 \) and \( g_2 \) on top of the shared encoder \( f \). Each classification head \( g_i \) maps the output of \( f \) to a \((k+1)\)-dimensional distribution, i.e., \( g_i[f(x)] \in \mathbb{R}^{k+1} \) (\( i = 1, 2 \)), and a classification loss is optimized on IND samples:

\[
L_{\text{cls}}^{IND} = \sum_{x \in \mathcal{D}_I} \frac{1}{2} \sum_{i=1}^{2} CE(l_p(x), g_i[f(x)]),
\]
in which $CE$ measures the cross-entropy between two distributions.

Besides optimizing $L_{cls}^{IND}$, a co-training process is implemented to refine $l_i(x)$ for each $x \in D_P$. Specifically, a soft label $\hat{l}_i(x)$ (or $\hat{l}_i^2(x)$) is produced by interpolating $l_i(x)$ with the prediction of one classification head $g_1$ (or $g_2$), and the resulting soft label is used to optimize another head $g_2$ (or $g_1$). Concretely, the following co-training loss is optimized:

$$L_{co}^{OOD} = \sum_{x \in D_P} \frac{1}{2} \sum_{j=1}^{2} \sum_{i=1}^{2} l_{ij}^{2}(x) CE(l_i^2(x), g_i[f(x)]),$$

$$l_i^2(x) = \beta \cdot l_i(x) + (1 - \beta) \cdot g_i[f(x)], (i = 1, 2),$$

where $0 \leq \beta \leq 1$ is a weight scalar. Different dropout masks are used in $g_1$ and $g_2$ to prompt the diversity required by co-training. Note that as indicated by Lee et al. (2013) and Chen et al. (2022), the co-training loss $L_{co}^{OOD}$ favors low density separation between classes, and thus it helps to enforce the low-density assumption when training $g_i$.

The overall training loss for our method is:

$$\mathcal{L} = \mathcal{L}_{ctr} + \mathcal{L}_{cls}^{IND} + \mathcal{L}_{co}^{OOD}$$

### 4.4 OOD Detection

In the inference phase, we directly use the averaged prediction of $g_1$ and $g_2$ to implement the OOD detector $g(y|x) \in R^{k+1}$.

$$g(y|x) = (g_1[f(x)] + g_2[f(x)])/2$$

Moreover, an adaptive decision boundary (ADB) is learnt on top of $g(y|x)$ to further reduce the open space risk (Zhou et al., 2019, Shu et al., 2021). Specifically, we follow the approach of Zhang et al. (2021) to obtain a central vector $c_i$ and a decision boundary scalar $b_i$ for each intent class $I_i \in I \cup \{I_{k+1}\}$. In the testing phase, the label $y$ for each input $x$ is obtained as:

$$y = \begin{cases} 
I_{k+1}, & \text{if } ||f(x) - c_i|| > b_i, \forall i \in \{1, \ldots, k+1\}, \\
\arg \max_{I_j \in \mathcal{X} \setminus \{I_{k+1}\}} g(I_j|x), & \text{otherwise}, 
\end{cases}$$

In this way, we can classify IND intents while rejecting OOD intent.

### 5 Experiments

#### 5.1 Datasets

Following most previous works (Zhang et al., 2021; Shu et al., 2021), we use three benchmark datasets:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Train</th>
<th>Valid</th>
<th>Test</th>
<th>#Intent</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLINC150</td>
<td>15,000</td>
<td>3,000</td>
<td>5,700</td>
<td>150</td>
</tr>
<tr>
<td>StackOverflow</td>
<td>12,000</td>
<td>2,000</td>
<td>6,000</td>
<td>20</td>
</tr>
<tr>
<td>Banking</td>
<td>9,003</td>
<td>1,000</td>
<td>3,080</td>
<td>77</td>
</tr>
</tbody>
</table>

Table 1: Dataset statistics.

**CLINC150** (Larson et al., 2019) contains 150 IND intents and one OOD intent. We follow Zhan et al. (2021) to group all samples from the OOD intent into the test set; **StackOverflow** (Xu et al., 2015) contains 20 classes with 1,000 samples in each class; **Banking** (Casanueva et al., 2020) contains 77 intents in the banking domain. Standard splits of above datasets is followed (See Table 1).

### 5.2 Implementation Details

Our encoder $f$ is implemented using BERT (Devlin et al., 2018) with a mean-pooling layer. The projection head $h$ and classification heads $g_1$ and $g_2$ are implemented as two-layer MLPs with the LeakyReLU activation (Xu et al., 2020b). The optimizer AdamW and Adam (Kingma and Ba, 2014) is used to finetune BERT and all the heads with a learning rate of 1e-5 and 1e-4, respectively. We use $\tau = 0.1$, $\alpha = 0.11$ and $\beta = 0.9$ in all experiments. All results reported in our paper are averages of 10 runs with different random seeds. See Appendix B for more implementation details. Note that ASouI only introduces little computational overhead compared to the vanilla BERT model (See Appendix D), and we detail how to choose important hyper-parameters for ASouI in Appendix E.

### 5.3 Experiment Setups and Baselines

Following (Zhang et al., 2021; Zhan et al., 2021; Shu et al., 2021), we randomly sample 25%, 50%, and 75% intents as the IND intents and regard all remaining intents as one OOD intent $I_{k+1}$. Note that in the training and validation process, we only use samples from the IND intents. Hyper-parameters are searched based on IND intent classification performances on validation sets.

To validate our claim that ASouI is independent of specific methods to produce $D_P$, we tested the performance of ASouI with four pseudo OOD sample generation approaches: 1. **Phrase Distortion (PD):** follows Shu et al. (2021) to generates OOD samples by distorting IND samples; 2. **Feature Mixup (FM):** follows Zhan et al. (2021) to produce OOD features using convex combinations of
IND features; 3. Latent Generation (LG): follows Zheng et al. (2020a) to decode pseudo OOD samples from a latent space; 4. Open-domain Sampling (OS): follows Zhan et al. (2021) to use sentences from other corpora as OOD samples. Each approach mentioned above associates with one of the four categories listed in §2.

Moreover, we also applied the above pseudo OOD sample generation approaches with the previous SOTA method that uses one-hot labeled pseudo OOD samples (Shu et al., 2021). Specifically, a \((k + 1)\)-way classifier is trained by optimizing the cross-entropy loss on \(D_I \cup D_P\) using one-hot labels, and the ADB approach presented in §4.4 is used to construct the OOD detector.

We also compared our method to other competitive OOD detection baselines: MSP: (Hendrycks and Gimpel, 2017) utilizes the maximum Softmax probability of a \(k\)-way classifier to detect OOD inputs; DOC: (Shu et al., 2017) employs \(k\) 1-vs-rest Sigmoid classifiers and use the maximum predictions to detect OOD intents; OpenMax: (Bendale and Boul, 2016) fits a Weibull distribution to the logits and re-calibrates the confidences with an Open-Max Layer; LMCL: (Lin and Xu, 2019) introduces a large margin cosine loss to maximize the decision margin and uses LOF as the OOD detector; ADB: (Zhang et al., 2021) learns an adaptive decision boundaries for OOD detection; Outlier: (Zhan et al., 2021) mixes convex interpolated outliers and open-domain outliers to train a \((k+1)\)-way classifier; SCL: (Zeng et al., 2021a) uses a supervised contrastive learning loss to separate IND and OOD features; GOT: (Ouyang et al., 2021) shapes an energy gap between IND and OOD samples. ODIST: (Shu et al., 2021) generates pseudo OOD samples with using a pre-trained language model.

For fair comparisons, all baselines are implemented with codes released by their authors, and use BERT as the backbone. For threshold-based baselines, 100 OOD samples are used in the validation to determine the thresholds used for testing. See Appendix C for more details about baselines.

### 5.4 Metrics
Following Zhang et al. (2021); Zhan et al. (2021); Shu et al. (2021), we use overall accuracy \(\text{Acc-All}\) and macro F1-score \(\text{F1-All}\) calculated over

---

**Table 2:** Performances of ASoul when combined with different OOD sample generation approaches. Best results among each setting are bolded. The best performing ASoul-based method significantly outperforms other baselines with \(p\)-value < 0.05 (\(t\)-test) in each setting.

<table>
<thead>
<tr>
<th>Methods</th>
<th>CLINC150</th>
<th>StackOverflow</th>
<th>Banking</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Acc-All</td>
<td>F1-All</td>
<td>F1-OOD</td>
</tr>
<tr>
<td>LG+Onehot</td>
<td>58.19</td>
<td>68.46</td>
<td>49.04</td>
</tr>
<tr>
<td>OS+Onehot</td>
<td>47.92</td>
<td>60.26</td>
<td>7.93</td>
</tr>
<tr>
<td>PD+Onehot</td>
<td>32.38</td>
<td>40.70</td>
<td>49.52</td>
</tr>
<tr>
<td>FM+Onehot</td>
<td>92.71</td>
<td>84.11</td>
<td>95.42</td>
</tr>
<tr>
<td>LG+ASoul</td>
<td>58.00</td>
<td>53.97</td>
<td>64.81</td>
</tr>
<tr>
<td>OS+ASoul</td>
<td>47.89</td>
<td>66.84</td>
<td>4.37</td>
</tr>
<tr>
<td>PD+ASoul</td>
<td>92.71</td>
<td>84.11</td>
<td>95.42</td>
</tr>
<tr>
<td>FM+ASoul</td>
<td>45.56</td>
<td>47.04</td>
<td>48.81</td>
</tr>
<tr>
<td>LG+OS</td>
<td>92.71</td>
<td>84.11</td>
<td>95.42</td>
</tr>
<tr>
<td>LG+PD</td>
<td>92.71</td>
<td>84.11</td>
<td>95.42</td>
</tr>
<tr>
<td>LG+FM</td>
<td>92.71</td>
<td>84.11</td>
<td>95.42</td>
</tr>
</tbody>
</table>

---

**Note:** Table includes the performance of ASoul combined with different OOD sample generation approaches. Best results among each setting are bolded. The best performing ASoul-based method significantly outperforms other baselines with \(p\)-value < 0.05 (\(t\)-test) in each setting.
We can observe large improvements of ASoul when labeled IND datasets are small (i.e., in 25% and 50% sets).

ASoul consistently outperforms FM+Onehot across all intents (IND and OOD intents) to evaluate the OOD detection performance. We also calculate macro F1-scores over IND intents (F1-IND) and OOD intent (F1-OOD) to evaluate fine-grained performances.

### 5.5 Results

Table 2 shows the OOD detection performance associated with different pseudo OOD sample generation approaches. Specifically, results marked with “ASoul” measures the performance of our method, while results marked with “Onehot” correspond to the performance of the previous SOTA method (Shu et al., 2021) that uses one-hot labeled samples. We can observe that: 1. ASoul consistently outperforms its one-hot labeled counterpart with large margins. This validates our claim that ASoul can be used to improve the OOD detection performance with different pseudo OOD sample generation approaches; 2. “hard” pseudo OOD samples yield by FM lead to sub-optimal performance when assigned with one-hot labels (i.e., FM+Onehot generally under-performs PD+Onehot), while it achieves the best performance when combined with ASoul. This demonstrates that assigning one-hot labels to hard pseudo OOD samples introduces noise to the training process and ASoul helps to alleviate these noises. 3. Although OOD samples yielded by the open-domain sampling approach are usually disjoint from the training task, they still benefit from ASoul. We suppose this is because the soft labels produced by ASoul prevent the OOD detector from becoming over-confident, which is important to improve the OOD detection performance.

Table 3 shows the performance of all baselines and our best method FM+ASoul. It can be seen that FM+ASoul significantly outperforms all baselines and achieve SOTA results on all three datasets. This validates the effectiveness of ASoul in improving the OOD detection performance. We can also observe large improvements of ASoul when labeled IND datasets are small (i.e., in 25% and 50% sets).
tings). This demonstrates the potential of ASoul to be applied in practical scenarios, particularly in the early phases of the development that we usually need to handle a large number of OOD inputs with limited IND intents (Zhan et al., 2021).

5.6 Ablation Study

Ablation studies were performed to verify the effect of each component in ASoul: We tested following variants: 1. **ASoul-CT** removes the co-training framework, i.e., only one classification head $g_1$ is implemented without the co-training process. In this variant, the loss shown in Eq.6 is optimized by moving $g_2$ and setting $\beta = 1$ in Eq.5. 2. **ASoul-GS** removes the graph-smoothed labels, i.e., the embedding graph is not constructed. In this variant, losses shown in Eq.4 and 5 are optimized and $l_g(x)$ in Eq.5 is replaced with the one-hot prior label $l_p(x)$. 3. **USoul** employs uniformly distributed soft labels for samples in $D_P$. In this variant, the soft label $l_s^i(x)$ in Eq.5 is obtained by uniformly reallocating a small portion of OOD probability to OOD intents. 4. **KnowD** implements a knowledge distillation process to obtain soft labels, i.e., a $k$-way IND intent classifier is first trained on $D_I$ and its predictions are interpolated with the one-hot OOD label to obtain the soft label $l_s^i(x)$ in Eq.5.

All above variants are tested with two approaches to produce $D_P$: PD and FM. Results in Table 4 indicate that our method outperforms all ablation models. We can further observe that: 1. soft-labels obtained using other approaches degenerate the model performance by a large margin. This shows the effectiveness of the soft labels produced by ASoul. 2. graph-smoothed labels bring the largest improvement compared to other components. This further proves the importance of modeling semantic connections between OOD samples and IND intents.

5.7 Feature Visualization

To further demonstrate the effectiveness of ASoul, we visualized the features learnt in the penultimate layer of OOD detectors that are trained using one-hot labels or soft labels. We use the best performing pseudo OOD samples generation approach (i.e., FM) in this analysis. Results shown in Figure 4 demonstrate that soft labels produced by ASoul help the OOD detector learn better representations compared to one-hot labels. The learnt feature space is smoother and representations for IND and OOD samples are more separable. This validates our claim that ASoul helps to conform to the smoothness assumption and improves the OOD detection performance.

6 Conclusion

In this paper, we first analyze the limitation of existing OOD detection approaches that use one-hot labeled pseudo OOD samples. Then we propose a method ASoul that can estimate soft labels for given pseudo OOD samples and use these soft labels to train better OOD detectors. An embedding space is constructed to produce graph-smoothed labels to capture the semantic connections between OOD samples and IND intents. A co-training framework further refines these graph-smoothed labels. Experiments demonstrate that our method can be combined with different pseudo OOD sample generation approaches, and it helps achieve SOTA results on three benchmark datasets. In the future, we plan to apply our method in other tasks, such as Text-to-SQL parsers (Hui et al., 2021; Wang et al., 2022; Qin et al., 2022) or lifelong learning (Dai et al., 2022).

Limitations

We identify the major limitation of this work is its input modality. Specifically, our method is limited to textual inputs and ignores inputs in other modalities such as vision, audio, or robotic features. These modalities provide valuable information that can be used to better OOD detectors. Fortunately, with the help of multi-modal pre-training models (Radford et al., 2021; Zheng et al., 2022), we can obtain robust features well aligned across different modalities. In future works, we will try to model multi-modal contexts for OOD detection and explore better pseudo OOD sample generation approaches.
Another limitation of this work is the pre-training model used in experiments: a model pre-trained on dialogue corpora is expected to yield better performance (He et al., 2022c,a,b; Zhou et al., 2021b; Wang et al., 2020; Zheng et al., 2020b). Moreover, it is reported that better OOD detection performance can be obtained if we can extract more robust features for IND tasks (Vaze et al., 2021). Our method can be readily applied to other feature extractors that are better performed on dialogues.

**Ethics Statement**

This work does not present any direct ethical issues. In the proposed work, we seek to develop a general method for OOD intent detection, and we believe this study leads to intellectual merits that benefit from a reliable application of NLU models. All experiments are conducted on open datasets.

**References**


This appendix shows more pseudo OOD samples that are generated using existing approaches.

Besides Figure 2, we also visualize pseudo OOD samples produced by Zhan et al. (2021) on the CLINC150 (Larson et al., 2019) dataset (Figure 5) and the StackOverflow (Xu et al., 2015) dataset (Figure 6), when 25% intents are randomly selected as IND intents. Specifically, Zhan et al. (2021) proposes to generate features of pseudo OOD samples by mixing up IND features. Pseudo OOD samples we demonstrate in this analysis are obtained using the code released by Zhan et al. (2021). As shown in Figure 5 and 6, some pseudo OOD samples fall into the cluster of IND intents, and thus we argue it is improper to assign one-hot OOD labels to these samples.

Furthermore, we demonstrate more cases of pseudo OOD samples generated by the method proposed by Shu et al. (2021) on the CLINC150 (Table 5), StackOverflow (Table 6), and Banking (Table 7) datasets. Specifically, these pseudo OOD samples are generated by replacing phrases in IND samples and use a pre-trained language model to filter these samples. As shown in Table 5, 6, and 7, some of the generated pseudo OOD samples carry IND intents since the replaced phrase may be an synonyms of the original phrase.

B More Implementation Details

We use the BERT model (bert-base-uncased) provided in the Huggingface’s Transformers library (Wolf et al., 2020) to implement $f$. Following (Zhang et al., 2021), we add an averaging-pooling layer on top of BERT to obtain the representation of each input utterance. The projection head $h$, classification heads $g_1$ and $g_2$ are implemented as a two-layer MLP with the LeakyReLU activation (Xu et al., 2020b), where the feature dimension is 1024 and the projection dimension is 128. Following (Zhan et al., 2021), We use AdamW (Kingma and Ba, 2014) to fine-tune BERT using a learning rate of $1e^{-5}$ and Adam (Wolf et al., 2019) to train the MLP heads using a learning rate of $1e^{-4}$ with early stopping. When learning the adaptive decision boundaries (Zhang et al., 2021), the trained model is fixed and we used a learning rate of 0.05. We tried batch size of $\{100, 200\}$ for IND samples and $\{100, 500, 600, 800\}$ for OOD samples. All hyper-parameters are tuned according to the classification performance over the IND samples on the validation set. We find that $\tau = \tau = 0.1, \alpha = 0.11$ and $\beta = 0.9$ work well with all datasets. We use a dropout rate 0.6 for the two classification heads. Each result is an average of 10 runs with different random seeds, and each run is stopped when we reach a plateau on the validation loss. ALL
Table 5: Case study of generated OOD samples with ODIST on the CLINC150 dataset.

<table>
<thead>
<tr>
<th>Original Intent</th>
<th>Original Utterance before Distortion</th>
<th>Generated Pseudo OOD Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>translate</td>
<td>how do you say hi in french</td>
<td>how do you spell hi in french</td>
</tr>
<tr>
<td>translate</td>
<td>how do you say please in arabic</td>
<td>how do you say please in Spanish?</td>
</tr>
<tr>
<td>transfer</td>
<td>transfer $500 from my checking to my savings</td>
<td>transfer $50 from my checking to my savings</td>
</tr>
<tr>
<td>transfer</td>
<td>can you transfer $5 from savings to checking</td>
<td>can you transfer your $5,000 from savings to checking</td>
</tr>
<tr>
<td>insurance_change</td>
<td>what do i do for new insurance</td>
<td>what do you do for new insurance</td>
</tr>
<tr>
<td>insurance_change</td>
<td>i would like to switch my insurance plan</td>
<td>i had to switch my insurance plan</td>
</tr>
<tr>
<td>travel_alert</td>
<td>is it safe to travel to argentina</td>
<td>is it safe to travel to Europe?</td>
</tr>
<tr>
<td>travel_alert</td>
<td>tell me about any travel alerts issued for germany</td>
<td>Read more about any travel alerts issued for germany</td>
</tr>
<tr>
<td>fun_fact</td>
<td>can you tell me something i don’t know about banks</td>
<td>can you tell me something i don’t know about you</td>
</tr>
<tr>
<td>fun_fact</td>
<td>can you tell me fun facts about lighthouses</td>
<td>can you tell me fun facts about me?</td>
</tr>
</tbody>
</table>

Table 6: Case study of generated OOD samples with ODIST on the StackOverflow dataset.

<table>
<thead>
<tr>
<th>Original Intent</th>
<th>Original Utterance before Distortion</th>
<th>Generated Pseudo OOD Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>wordpress</td>
<td>Multiple loop working, function inside isn’t</td>
<td>Multiple looping function inside is n’t</td>
</tr>
<tr>
<td>wordpress</td>
<td>Plugin to avoid share username in Wordpress</td>
<td>Plugin to share username in Wordpress</td>
</tr>
<tr>
<td>visual-studio</td>
<td>Visual Studio Find and Replace Variables</td>
<td>Visual Studio Find and Destroy</td>
</tr>
<tr>
<td>visual-studio</td>
<td>Number of Classes in a Visual Studio Solution</td>
<td>Number of Users in a Visual Studio Solution</td>
</tr>
<tr>
<td>svn</td>
<td>how to update a file in svn?</td>
<td>how do you open a file in svn?</td>
</tr>
<tr>
<td>svn</td>
<td>Revert a whole directory in tortoise svn?</td>
<td>What is a whole directory in tortoise svn?</td>
</tr>
<tr>
<td>spring</td>
<td>Problem with Autowiring &amp; No unique bean</td>
<td>Problem with Autowiring &amp; the unique bean</td>
</tr>
<tr>
<td>spring</td>
<td>Creating temporary JMS jms topic in Spring</td>
<td>Creating Your Own Home in Spring</td>
</tr>
<tr>
<td>scala</td>
<td>Can Scala survive without corporate backing?</td>
<td>Can Scala get corporate backing?</td>
</tr>
<tr>
<td>scala</td>
<td>What is are differences between Int and Integer in Scala?</td>
<td>What is are differences between Int and Int++ in Scala?</td>
</tr>
</tbody>
</table>

Table 6: Case study of generated OOD samples with ODIST on the StackOverflow dataset.
experiments were conducted in the Nvidia Tesla V100-SXM2 GPU with 32G graphical memory. Our model contains 112.06M model parameters.

C More Details about Baselines

Baseline results (MSP, DOC, OpenMax, LMCL, and ADB) are copied from (Zhang et al., 2021). Results of the baseline Outlier are copied from (Zhan et al., 2021). Results of the baseline ODIST are copied from (Shu et al., 2021). For above mentioned baselines, we also re-implement their methods using their release codes. The results reproduced by our experiments match the results reported in their original paper. So we copied the highest reported results of these baselines from previously published papers. Significant tests between our method and all these baselines are carried out based on our implementations. We get the baseline results (SCI and GOT) by running their released codes, and use 100 OOD samples in the validation to determine the thresholds for testing.

D Computational Cost Analysis

We compare the training cost of our method when using one-hot labels or soft labels produced by ASoul. Pseudo OOD samples used in this analysis are generated using the best performing method FM, and we use the CLINC150 dataset for this analysis. As shown in Table 8, ASoul only introduces marginal parameter overhead for the projection head and the classification head. We can also observe that using ASoul only introduces little time overhead compared to the one-hot labeling approach.

<table>
<thead>
<tr>
<th>Methods</th>
<th>#para.</th>
<th>25%</th>
<th>50%</th>
<th>75%</th>
</tr>
</thead>
<tbody>
<tr>
<td>FM+Onehot</td>
<td>111.36M</td>
<td>7.97s</td>
<td>15.76s</td>
<td>23.40s</td>
</tr>
<tr>
<td>FM+Asoul</td>
<td>112.06M</td>
<td>8.57s</td>
<td>16.85s</td>
<td>25.06s</td>
</tr>
</tbody>
</table>

Table 8: Number of parameters (Million) and average training time for each epoch (seconds) on the CLINC150 dataset.

E Effect of Hyper-parameters

We analyzed the most important hyper-parameters of ASoul: temperature $\tau$ in graph-based smoothing and dropout rate to the classification heads in co-training. We conduct experiments to show their effects on the CLINC150 dataset under the 25% setting. We use the best performing pseudo OOD sample generation approach (i.e., FM) in this analysis.

Temperature: We set $\tau$ to \{0.1, 0.5, 1, 5, 10, 15, 20\} respectively, and demonstrate the performance change. Note that small $\tau$ makes distribution over
the embedding graph more shape (concentrating on nearest neighbors), while large \( \tau \) forms smooth distributions.

Results are shown in Figure 7 (left). With the increase in temperature, the OOD detection performance tends to decrease. \( \tau = 0.1 \) achieves the highest F1-ALL score of 84.11%. This suggests that a small temperature makes ASoul focus more on neighbors and gain better performance.

**Dropout Rate:** We compare the performance of dropout rates to the classification heads by adjusting the rate from 0 to 0.7 with an interval of 0.1.

Results are shown in Figure 7 (right). The performance first increases and then decreases as the dropout rate increases. In the begging phase, using a high dropout rate introduces more diversity required by co-training, and thus the OOD detection performance improves. However, using a higher dropout rate introduces much noise to the co-training process, and thus downgrades the OOD detection performance.

**F More Evaluation Metrics**

We also calculate micro F1-scores over all intents (IND and OOD intents) for our best-performing method FM+ASoul and one of our strongest baselines Outlier on the CLINC150 dataset. As shown in Table 9, FM+ASoul still outperforms the baseline on the micro F1-score.

<table>
<thead>
<tr>
<th>Methods</th>
<th>25%</th>
<th>50%</th>
<th>75%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outlier</td>
<td>91.68</td>
<td>88.23</td>
<td>88.46</td>
</tr>
<tr>
<td>FM+Asoul</td>
<td><strong>93.30</strong></td>
<td><strong>90.36</strong></td>
<td><strong>89.98</strong></td>
</tr>
</tbody>
</table>

Table 9: Performances of Outlier and FM+Asoul on the CLINC150 dataset under the metric of micro F1-score over all intents (IND and OOD intents).