Joint Learning of Domain Classification and Out-of-Domain Detection with Dynamic Class Weighting for Satisficing False Acceptance Rates

Joo-Kyung Kim and Young-Bum Kim

Amazon Alexa
{jookyk, youngbum}@amazon.com

Abstract

In domain classification for spoken dialog systems, correct detection of out-of-domain (OOD) utterances is crucial because it reduces confusion and unnecessary interaction costs between users and the systems. Previous work usually utilizes OOD detectors that are trained separately from in-domain (IND) classifiers, and confidence thresholding for OOD detection given target evaluation scores. In this paper, we introduce a neural joint learning model for domain classification and OOD detection, where dynamic class weighting is used during the model training to satisfy a given OOD false acceptance rate (FAR) while maximizing the domain classification accuracy. Evaluating on two domain classification tasks for the utterances from a large spoken dialogue system, we show that our approach significantly improves the domain classification performance with satisficing given target FARs.

Index Terms: domain classification, out-of-domain detection, false acceptance rate, dynamic class weighting

1. Introduction

Domain classification is one of the three major components of spoken language understanding along with intent detection and slot filling [1]. Errors made by domain classifiers are more critical than errors by the other components because the domain classification errors tend to be propagated to completely incorrect system actions or responses. Since recent spoken dialog systems such as Amazon Alexa, Google Assistant, Microsoft Cortana, and Apple Siri deal with a wide variety of scenarios [2, 3], correct domain classification of a user’s utterance into one of the supported domains or out-of-domain (OOD) is becoming more complex and important.

Domain classifiers are usually trained focusing on maximizing a single evaluation metric such as classification accuracy and F-score. In real spoken dialog systems, however, correctly detecting OOD utterances is crucial because spoken dialog systems are prone to receive various OOD utterances such as unactionable utterances, ungrammatical utterances, and those with severe ASR errors. Therefore, misclassifying OOD utterances as in-domain (IND) causes confusion and unnecessary interaction costs between users and the dialog systems. To reduce the OOD misclassification of domain classifiers, false acceptance rate (FAR), which is 1 - OOD recall, is often regarded as a satisficing metric that must be below a predefined value while false rejection rate (FRR), which is 1 - IND recall, is considered relatively less important.

Previous approaches for OOD detection usually train OOD detectors separately or on top of IND classifiers [4, 5, 6, 7]. In these methods, OOD detectors are trained only to identify whether the given utterances are OOD or not regardless the classification of IND utterances. Also, the methods are evaluated either based on IND recall and OOD recall [5] or Equal Error Rate (ERR), where thresholding is used to match FAR and FRR to be the same [4, 6]. Consequently, they do not specifically focus on keeping FARs to be low.

Dealing with those issues, we introduce a joint learning model of IND classification and OOD detection. Joint learning models have been shown effective for various spoken language understanding tasks. For example, joint training of intent detection and slot-filling [8]; joint training of all the three SLU components [9, 10], and joint training of multiple domains [11, 12, 13] have been shown synergistic since the jointly trained components are highly related to each other. Our model jointly trains a multi-class classifier for the domain classification and a binary classifier for the OOD detection on top of a bidirectional Long Short-Term Memory (BiLSTM) layer [14] for the utterance representations. Within this joint architecture, IND classification and OOD detection are helpful to each other by sharing underlying vector representations. In addition, we use dynamic class weighting, where we adjust the class weights for the IND and OOD loss functions to satisfy the FAR on the development set for each epoch. With dynamic class weighting, we first focus on the FAR as a satisficing metric that must be equal or lower than a predefined target value and then the IND accuracy as an optimizing metric.

Evaluating on two datasets collected by Amazon Alexa, we show that our joint learning model, which aims to satisfy the FAR and maximize the overall classification accuracy with dynamic class weighting, significantly improves domain classification performance given the two metrics.

2. Satisficing false acceptance rates

Our objective in this paper is having FAR as a satisficing metric and the domain classification accuracy as an optimizing metric. This is relevant to addressing class imbalance or unequal class cost cases in classification, where techniques such as oversampling, undersampling, SMOTE, class weighting, and threshold-moving are commonly used [15, 16, 17, 18, 19]. However, our objective is different from class imbalance or unequal class costs because of the following reasons:

- We have two metrics (FAR and accuracy) to optimize rather than one.
- FAR must be satisfied to be equal or below a predefined target value.
- Proper oversampling rates or class weights for satisficing a given FAR is difficult to be decided in advance since they would be substantially different for different datasets.

We formulate the objective as a non-differentiable con-
strained optimization problem as follows:

\[
\text{max accuracy (D) \ subject \ to \ } \text{FAR (D) } \leq T, \quad (1)
\]

where \( D \) is an evaluated dataset and \( T \) is a target FAR. In Section 3, we describe our model and formulate a differentiable surrogate loss function to address the objective.

### 3. Model

Figure 1 shows the overall architecture of the proposed joint model of domain classification and OOD detection.

#### 3.1. Word representation

In order to leverage character-level vector representations, we use both character embeddings and word embeddings to represent each word [20]. Let \( C \) and \( W \) denote the set of characters and the set of words, respectively. Let \( \oplus \) denote the vector concatenation operator. We formulate LSTM [21] as a function \( \phi : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^d \) that takes an input vector \( x \) and a state vector \( h \) to output a new state vector \( h' = \phi(x, h) \).

The model parameters associated with the word representations are:

- **Char embedding**: \( e_c \in \mathbb{R}^{25} \) for each \( c \in C \)
- **Char LSTMs**: \( \phi^C, \phi^b : \mathbb{R}^{25} \times \mathbb{R}^{25} \rightarrow \mathbb{R}^{25} \)
- **Word embedding**: \( e_w \in \mathbb{R}^{100} \) for each \( w \in W \)

Let \((w_1, \ldots, w_n)\) denote a word sequence where \( w_i \) has a character \( w_i(j) \in C \) at position \( j \). The vector representation of the \( i \)-th word, \( v_i \in \mathbb{R}^{250} \), is obtained by the concatenation of the both ends of the character LSTM outputs and the \( i \)-th word vector \( e_{w_i} \).

It is formulated as follows:

\[
\begin{align*}
   f^c_i &= \phi^c_i \left( e_{w_i(j)}, f^c_{i-1} \right) \quad \forall j = 1 \ldots |w_i| \\
   b^c_i &= \phi^b_i \left( e_{w_i(j)}, b^c_{i+1} \right) \quad \forall j = |w_i| \ldots 1 \\
   v_i &= f^c_{|w_i|} \oplus b^c_i \oplus e_{w_i}
\end{align*}
\]

where \( f^c_{|w_i|}, b^c_i \) denote the forward LSTM output for the last character and the backward LSTM output for the first character, respectively.

#### 3.2. Utterance representation

We encode the word vector sequence \((v_1, \ldots, v_n)\) with a BiLSTM for each \( i = 1, \ldots, n \):°

- **Word LSTMs**: \( \phi_i^W, \phi_i^b : \mathbb{R}^{150} \times \mathbb{R}^{100} \rightarrow \mathbb{R}^{100} \)

\[
\begin{align*}
   f^W_i &= \phi_i^W \left( v_i, f^W_{i-1} \right) \quad \forall i = 1 \ldots n \\
   b^W_i &= \phi_i^b \left( v_i, b^W_{i+1} \right) \quad \forall i = n \ldots 1
\end{align*}
\]

Then, in a similar way to obtaining the word representations from the character BiLSTMs, we concatenate the last outputs of both word LSTMs to represent the whole utterance as a single 200 dimensional vector:

\[
   u = f^W_n \oplus b^W_1.
\]

We also evaluate two other utterance representation methods, which are word vector summation \( u = \sum_{i=1}^n v_i \) and convolutional neural networks (CNN).°

\[\text{\footnotesize{\textsuperscript{1}}}\text{e}_{w_i} \text{~is pretrained with GloVe leveraging Wikipedia 2014 and Gigaword 5 corpora [22].}}\]

\[\text{\footnotesize{\textsuperscript{2}}}f^c_i, b^c_i, f^W_i, b^W_i \text{ are randomly initialized vectors.}}\]

\[\text{\footnotesize{\textsuperscript{3}}}\text{In a similar method to [23], we use three convolution filters whose}}\]
3.3. Domain classification
On top of the utterance vector $u$, we use a feed-forward neural network $\phi_d$ and softmax function to obtain the probability distribution over the entire domains as:

$$d = \text{softmax}(\phi_d(u)).$$

The loss function for the domain prediction is formulated as the cross-entropy between the label and the probability distribution of an utterance:

$$L_D(d) = -d \log \hat{d},$$

where $\hat{d}$ is the one-hot representation of the ground-truth domain of the current utterance.

3.4. OOD detection
Along with the domain classifier, we jointly train an OOD detector, which predicts whether the current utterance is IND or OOD. We use a feed-forward network $\phi_o$ as did for the domain classification:

$$o = \text{softmax}(\phi_o(u)).$$

The loss function for OOD detection is formulated as:

$$L_O(o) = -\hat{o} \log o,$$

where $\hat{o}$ is a two dimensional one-hot vector representing whether the current utterance is IND or OOD.

3.5. Joint loss function
A loss function for combining domain classification and OOD detection is formulated as follows:

$$L_J(o) = L_D(o) + \alpha L_O(o),$$

where $\alpha$ is a hyperparameter that controls the degree of the influence from the binary OOD detector. We show results on different $\alpha$ values in Section 4.2.

3.6. Dynamic class weighting
The final loss function, which approximately optimizes Equation 1, is as follows:

$$L(D) = (2 - \lambda) \sum_{d_k \in D_{IND}} L_J(d_k) + \lambda \sum_{d_l \in D_{OOD}} L_J(d_l),$$

where $D_{IND}$ is the set of utterances with IND ground-truths, $D_{OOD}$ is the utterance set with OOD ground-truths, $L_J$ is the joint loss function in Section 3.5, and $\lambda$ is a parameter deciding the class weights for IND domains and the OOD. Here, $\sum_{d_k \in D_{IND}} L_J(d_k)$ and $\sum_{d_l \in D_{OOD}} L_J(d_l)$ are surrogate loss functions for maximizing the IND classification and satisficing the FAR, respectively. This formulation uses $2 - \lambda$ and $\lambda$ as the class weights for IND and OOD, respectively. The main issue of Equation 5 is that we cannot predetermine $\lambda$ as aforementioned.

To obtain a proper $\lambda$, we introduce dynamic class weighting for OOD, where $\lambda$ is changed during the training so that the FAR on the development set is satisfied with minimal $\lambda$ increase for OOD. We initialize $\lambda$ to 1 so that the class weights for both IND and OOD are 1 in the beginning. At the end of each training epoch, we calculate the FAR on the development set. If the current FAR does not satisfice the target FAR, we add $\gamma$ to current $\lambda$. Oppositely, if the target FAR is satisficed, we subtract $\gamma$ from current $\lambda$. When adding and subtracting $\gamma$, we limit the $\lambda$ value to be less than 2 and larger than 0. In our work, we initialize $\gamma$ to 0.1. 6 To reduce fluctuations of $\lambda$ during the late epochs, we halve $\gamma$ each time when the target FAR is satisficed in the current epoch but not in the previous epoch. With this approach, we encourage the model to find the minimal class weight change that satisfices the FAR and then focuses on the overall classification accuracy.

4. Experiments
We have conducted a series of experiments to evaluate the proposed method on datasets obtained from real usage data in Amazon Alexa with two domain classification tasks.

4.1. Datasets
We evaluate our models on two domain classification tasks from different data sources: (1) utterances from 21 Alexa domains, (2) utterances from frequently used 1,500 skills out of more than 40,000 skills. 7 For both cases, we use randomly sampled unique utterances that are collected and annotated from the real user logs. The average utterance lengths are 5.96 and 5.68 for the 21 domains and the 1,500 skills, respectively. Table 1 shows the statistic of the datasets.

<table>
<thead>
<tr>
<th></th>
<th>21 domain dataset</th>
<th>1,500 skill dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IND</td>
<td>OOD</td>
</tr>
<tr>
<td>Train</td>
<td>712k</td>
<td>255k</td>
</tr>
<tr>
<td>Dev</td>
<td>112k</td>
<td>17k</td>
</tr>
<tr>
<td>Test</td>
<td>112k</td>
<td>21k</td>
</tr>
</tbody>
</table>

Table 1: The numbers of the utterances in the two datasets: 21 Alexa domains and 1,500 Alexa skills.

6 We also tried different values but there were no significant differences in the experiment results.

7 In Amazon Alexa, a skill is a domain developed by third-party developers [25, 26].
Table 2: The test classification accuracies (%) of various models given different satisfying FARs on the 21 domain dataset. \( \alpha \) of Equation 4 is set to 0 for Separate case and 1 for all the other cases. \( \alpha \) is the coefficient for the OOD detector loss.

<table>
<thead>
<tr>
<th>Model</th>
<th>( \alpha )</th>
<th>6%</th>
<th>5.5%</th>
<th>5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Separate (BiLSTM)</td>
<td>0</td>
<td>80.27</td>
<td>89.98</td>
<td>88.33</td>
</tr>
<tr>
<td>Joint (WordVecSum)</td>
<td>1</td>
<td>82.61</td>
<td>89.89</td>
<td>88.31</td>
</tr>
<tr>
<td>Joint (CNN)</td>
<td>1</td>
<td>82.27</td>
<td>89.88</td>
<td>88.31</td>
</tr>
<tr>
<td>Joint (BiLSTM)</td>
<td>1</td>
<td>82.67</td>
<td>90.52</td>
<td>90.26</td>
</tr>
<tr>
<td>w/ dynamic, class weighting</td>
<td>0.005</td>
<td>90.71</td>
<td>90.53</td>
<td>90.38</td>
</tr>
<tr>
<td>w/ dynamic, class weighting</td>
<td>0.01</td>
<td>90.69</td>
<td>90.61</td>
<td>90.28</td>
</tr>
<tr>
<td>w/ dynamic, class weighting</td>
<td>0.1</td>
<td>90.63</td>
<td>90.53</td>
<td>90.30</td>
</tr>
</tbody>
</table>

4.2.2. 1,500 Alexa Skills

This task deals with utterance classification to either one of 1,500 skills or OOD. For example, the skills for “what does a peacock say” and “find me the recipe for world’s best lasagna” should be predicted as ZooKeeper and AllRecipes, respectively. The skills are significantly more diverse and less well defined than 21 Alexa domains, which makes the classification more challenging. In real spoken dialog systems, the classification performance can be further improved by leveraging various contextual information [32, 33, 34, 26]. However, they are beyond the scope of this paper, and we leave the evaluation of our models on such reranking systems as future work.

On this task, when there is no target FAR, the accuracy and the FAR of the Joint (BiLSTM) model with \( \alpha = 0 \) are 80.65% and 3.62%, respectively.

Therefore, we have evaluated our models on lower target FARs, 2%, 1.5%, and 1%. Table 3 shows the results of our proposed models. Overall, similarly to the results of 21 Alexa Domains, we can see that Joint models are better than Separate model, using BiLSTM outperforms using word vector summation or CNN for the utterance representations, and Joint with dynamic class weighting models show significantly better performance than other models.

5. Conclusion

We have introduced a joint learning model of domain classification and OOD detection utilizing dynamic class weighting to satisfice a target FAR and then maximize the overall classification accuracy. Evaluating on two domain classification tasks for the utterances from Amazon Alexa, we have shown that our proposed joint learning models with dynamic class weighting is more effective than the models with separate learning of domain classification and OOD detection or those trained to optimize a single metric when we have FAR as a satisficing metric and accuracy as an optimizing metric.

6. References


