Confidence Calibration for Intent Detection via Hyperspherical Space and Rebalanced Accuracy-Uncertainty Loss

Yantao Gong\textsuperscript{1,2,3}, Cao Liu\textsuperscript{1}, Fan Yang\textsuperscript{1}, Xunliang Cai\textsuperscript{1}, Guanglu Wan\textsuperscript{1}, Jiansong Chen\textsuperscript{1}, Weipeng Zhang\textsuperscript{1}, Houfeng Wang\textsuperscript{3,†}

\textsuperscript{1} Meituan

\textsuperscript{2} School of Software and Microelectronics, Peking University

\textsuperscript{3} MOE Key Lab of Computational Linguistics, Peking University

\{gongyt, wanghf\}@pku.edu.cn \{liucao, yangfan79, caixunliang, wanguanglu, chenjiansong, zhangweipeng02\}@meituan.com

Abstract

Data-driven methods have achieved notable performance on intent detection, which is a task to comprehend user queries. Nonetheless, they are controversial for over-confident predictions. In some scenarios, users do not only care about the accuracy but also the confidence of model. Unfortunately, mainstream neural networks are poorly calibrated, with a large gap between accuracy and confidence. To handle this problem defined as confidence calibration, we propose a model using the hyperspherical space and rebalanced accuracy-uncertainty loss. Specifically, we project the label vector onto hyperspherical space uniformly to generate a dense label representation matrix, which mitigates over-confident predictions due to overfitting sparse one-hot label matrix. Besides, we rebalance samples of different accuracy and uncertainty to better guide model training. Experiments on the open datasets verify that our model outperforms the existing calibration methods and achieves a significant improvement on the calibration metric.

Introduction

Intent detection is a crucial portion in comprehending user queries, which generally predicts intent tags by semantic classification (Brenes, Gayo-Avello, and Pérez-González 2009; Qin et al. 2020). Therefore, it is widely used in many NLP applications, such as search, task-based dialogue, and other fields (Zhang and Wang 2016; Larson et al. 2019; Casanueva et al. 2020).

In recent years, data-driven methods develop rapidly and become a primary trend of intent detection. However, they are highly criticized for over-confident predictions (Niculescu-Mizil and Caruana 2005; Nguyen, Yosinski, and Clune 2015; Pereyra et al. 2017; Li, Dasarathy, and Berisha 2020). As shown in Figure 1(a), there is a serious phenomenon that the prediction confidence (i.e. probability associated with the predicted label) of samples is very high even if the samples are misclassified. For example, when the confidence is in [0.9-1], the proportion of misclassified samples reaches 35.67%. Besides, the average confidence (90.39%) is evidently over the accuracy (56.17%).

One of the effective solutions to deal with the aforementioned problem is confidence calibration. A perfectly calibrated model is supposed to output average confidence equal to the accuracy (Kong et al. 2020; Küppers et al. 2020). Unfortunately, due to over-parameterization and overfitting of the conventional methods, mainstream neural networks are poorly calibrated (Krishnan, Tickoo, and Tickoo 2020; Wang et al. 2020b; Schwaiger et al. 2021; Enomoto and Eda 2021). As demonstrated in Figure 1, “Gap” means the discrepancy between the average confidence and accuracy. The larger the “Gap” as, the worse the model is calibrated. Model without calibration, indicated in Figure 1(b), easily faces under-estimation problem when confidence is less than 0.2 and over-estimation problem when confidence is more than 0.4. Therefore, it owns a higher expected calibration error (ECE, calibration metric, more details in Experiments Section) than perfectly calibrated model.

To handle the confidence calibration problem, researchers have proposed numerous works (Nguyen and O’Connor 2015; Szegedy et al. 2016a; Müller, Kornblith, and Hinton 2019). One primary calibration approach acts on the post-processing stage. Guo et al. (2017) provide temperature scaling, which learns a single parameter from the development dataset to rescale all the logit before transmitting to softmax.
To validate the effectiveness of our model, we conduct abundant experiments on the three open datasets. Empirical results demonstrate that our model achieves evident improvements compared with the SOTA. Specifically, F1 increases on all the datasets with the calibration metric (ECE) drops down 10.50% on average. On the TNEWS dataset, the ECE achieves an obvious amelioration of 29.67% and the F1 obtains 21% promotion. Furthermore, our model acquires better performance among the existing methods on noisy data and low-frequency labels.

To sum up, our contributions are as follows:

1. We uniformly project the label vectors onto the hyperspherical space to obtain a denser representation matrix, which mitigates the model to overfit the sparse one-hot label matrix and generate over-confident predictions.

2. We rebalance the accuracy and uncertainty of samples and optimize the accurate samples with low uncertainty and inaccurate samples with high uncertainty separately by RAU loss to provide better guidance in the training process.

3. The experimental results demonstrate that our model gains an advantage over the SOTA, not only in the F1 but also in the confidence calibration metric. Moreover, we obtain noteworthy performance on noisy data and low-frequency labels.

Related Work

Intent Detection. Intent is the semantic purpose of a query, which is generated by users (Xu and Sarikaya 2013; Wang, Tang, and He 2018). As a matter of fact, the essence of intent detection is text classification (Brenes, Gayo-Avello, and Pérez-González 2009; Mehri, Eric, and Hakkani-Tur 2020; Chatterjee and Sengupta 2020). After training on the dataset with ground-truth labels, the model attempts to predict the intent of query within the existing intent set. There have been plenty of researches on conventional neural network methods in the last few decades (Xu and Sarikaya 2013; Liu and Lane 2016; Zhang et al. 2019; Haibong et al. 2019; Wang et al. 2020a; Gerz et al. 2021). During recent years, with the rapid development of computing power, pre-trained models such as BERT (Devlin et al. 2018) are employed for intent detection frequently (Castellucci et al. 2019; He et al. 2019; Zhang, Zhang, and Chen 2019; Athiwaratkun et al. 2020; Gong et al. 2021).

Confidence Calibration. Confidence calibration has a long history of research in statistical machine learning (Brier 1950; Griffin and Tversky 1992; Gneiting and Raftery 2007). In the past several years, one major calibration methods fall into the post-processing stage (Platt 1999; Zadrozny and Elkan 2001; Kumar, Liang, and Ma 2019; Zhang, Kailkhura, and Han 2020; Rahimi et al. 2020). For example, Guo et al. (2017) propose the temperature scaling. The trained model learns a single calibration scale from the development set. Another main calibration approaches try to optimize a function that represents the difference of average confidence and accuracy (Kumar, Sarawagi, and Jain 2018;
Kull et al. 2019; Mukhoti et al. 2020; Gupta et al. 2020). For instance, Krishnan, Tickoo, and Tickoo (2020) devise a loss function to combine accuracy and uncertainty. Jung et al. (2020) come up with a method to minimize the distribution between predicted probability and empirical probability.

**Method**

**Model Overview**

As shown in Figure 3, we employ the hyperspherical space and rebalanced accuracy-uncertainty loss to process confidence calibration. First, we use a text encoder such as BERT to acquire the encoded vector of the input query. Next, through the hyperspherical encoder, we obtain the dense encoded matrix of the input labels, which alleviates the overconfident predictions caused by the overfitting of sparse one-hot label matrix. After that, we utilize the hyperspherical decoder to calculate the calibration direction matrix and calibration scale. Furthermore, we separate the samples according to their accuracy and uncertainty, rebalance the samples’ accuracy and uncertainty by the RAU loss. Through the above process, we acquire the output with calibration.

**Hyperspherical Space Calibration**

In this submodule, we introduce how to separate the hyperspherical space homogeneously and project label vectors onto the hyperspherical space to obtain a dense label matrix. For \( N \) queries \( \{Q_1, \ldots, Q_N\} \), the corresponding labels are \( \{T_1, \ldots, T_N\} \), where \( T_i \in C \), \( C = \{1, \ldots, K\} \) indicates the set of \( K \) label tags. We exploit the text encoder like BERT to extract the encoded vector \( E_i \) (\( H \) dimension) such as [CLS] of each input query \( Q_i \). The encoded vector matrix \( E \) of all the queries is calibrated in the hyperspherical decoder.

**Text Encoder**

Before the learning process, we separate the \( H \)-dimensional output space \( S^H \) into \( K \) subspaces uniformly, which has the same size as the label set \( C \). Then, we define the vector of the hyperspherical label as \( \{h_1, \ldots, h_i, \ldots, h_K\} \), corresponding to the \( K \) subspaces. In addition, the norm of each vector satisfies \( \|h_i\| = 1 \). The dimension of hyperspherical label vector is \( H \), which equals the dimension of encoded vector. The hyperspherical encoder encodes each input label to a dense hyperspherical label vector, which is utilized in the hyperspherical decoder for calibration.

Here comes the detail of uniformly projecting the label vectors onto hyperspherical space. For each label vector \( h_i \) in the hyperspherical space, it has \( K - 1 \) cosine distances between all the \( K - 1 \) label vectors except itself, and the max cosine distance among them is \( D_i \), defined as below:

\[
D_i = \max(d_{ij})
\]

where \( i, j \in C \) and \( i \neq j \). \( d_{ij} \) is the cosine distance between label vector \( h_i \) and \( h_j \). As our goal is to make the label vector uniformly distributed in the hyperspherical space, therefore, it is equivalent to the optimization problem that minimizes
the sum of the maximum cosine distance $D_i$ of each label vector, as the following modality:

$$\mathcal{L}_h = \min \frac{1}{K} \sum_{i=1}^{K} D_i$$

(2)

Furthermore, due to all the label vectors are unit vectors, the above formula can be converted to matrix multiplication, which speeds up the calculation, by the following equations:

$$\mathcal{L}_h = \min \frac{1}{K} \sum_{i=1}^{K} \max(Z_i),$$

(3)

$$Z = X \cdot X^T - 2I$$

where $X = [h_1, ..., h_i, ..., h_K]$ is the matrix of hyperspherical label vector. $I$ is the identity matrix. $Z_i$ is the $i^{th}$ row of $Z$. In order to avoid self-selection, $Z$ subtracts identity matrix $I$ twice.

**Hyperspherical Decoder** After acquiring the encoded query vector and the dense encoded hyperspherical label vector through hyperspherical encoder, we utilize the hyperspherical decoder to get the calibration direction and the calibration scale.

We perform the dot product of the encoded vector with each hyperspherical label vector to get the calibration direction matrix $\text{Cali}_D$, formulated as below:

$$\text{Cali}_D = E \cdot X^T$$

(4)

where $E \in \mathbb{R}^{N \times H}$ denotes the encoded vector matrix of all the queries. $X^T \in \mathbb{R}^{H \times K}$ is the transpose matrix of dense hyperspherical label vector. Then, we calculate the norm of label vector matrix as the calibration scale $\text{Cali}_S$, which is the scale parameter during the overall process, by using the following equation:

$$\text{Cali}_S = ||X||$$

(5)

Finally, we compute the calibrated new logit $L$ as below:

$$L = \text{Cali}_S \times \text{Cali}_D$$

(6)

where the calibration scale $\text{Cali}_S$ is an unidimensional variable and the calibration direction $\text{Cali}_D \in \mathbb{R}^{N \times K}$.

**Rebalanced Accuracy-Uncertainty Loss**

In this submodule, we design the rebalanced accuracy-uncertainty loss to optimize accurate and inaccurate samples separately. Whether a sample is considered as accurate depends on whether the predicted label of the sample $T'_i$ equals to the exact sample’s label $T_i$, so we define the confidence (probability of predicted label) of a single sample as $a_i$ in the following:

$$a_i = \begin{cases} \max(p_i), & \text{if } T'_i = T_i \\ 1 - \max(p_i), & \text{otherwise.} \end{cases}$$

(7)

where $p_i$ is the predicted probability after transmitting to softmax. Therefore, when the predictions are accurate the $a_i$ is close to 1, while it is close to 0 when inaccurate. As there is no ground truth evaluation of the uncertainty, we utilize the calculation method described in Krishnan, Tickoo, and Tickoo (2020) to get the uncertainty $u_i$ as follows:

$$u_i = -p_i \log p_i$$

(8)

Then, we set the uncertainty threshold as $u_0 \in [0, 1]$, which is a heuristic setting obtained through the average uncertainty of training samples from initial epochs. A sample is defined as certain when the uncertainty of it is lower than $u_0$. Otherwise, it’s defined as uncertain. Then, we divide the training samples into four sets $\{AC, AU, IC, IU\}$ separately, where $AC$ means Accurate-Certain, $AU$ means Accurate-Uncertain, $IC$ means Inaccurate-Certain, and $IU$ means Inaccurate-Uncertain.

Based on the assumption mentioned in Krishnan, Tickoo, and Tickoo (2020), a well-calibrated model provides a low uncertainty for accurate predictions while it provides a high uncertainty for inaccurate predictions. Therefore, the model with calibration is supposed to produce a higher $AVU \in [0, 1]$ measure. $AVU$ is computed by summing the number of $\{AC, IC\}$ two sets, and then divide the total number of $\{AC, AU, IC, IU\}$ four sets.

To make the $AVU$ function differentiable for neural network parameters, we devise the calculation methods like:

$$n_{AC} = \sum_{i \in \{T'_i = T_i \text{ and } u_i \leq u_0\}} a_i \odot (1 - \tan(u_i)),$$

$$n_{AU} = \sum_{i \in \{T'_i = T_i \text{ and } u_i > u_0\}} a_i \odot \tan(u_i),$$

$$n_{IC} = \sum_{i \in \{T'_i \neq T_i \text{ and } u_i \leq u_0\}} a_i \odot (1 - \tan(u_i)),$$

$$n_{IU} = \sum_{i \in \{T'_i \neq T_i \text{ and } u_i > u_0\}} a_i \odot \tan(u_i)$$

(9)

where $\odot$ is hadamard product. In addition, we step further on and rebalance the accuracy-uncertainty, which prompts the model to respectively optimize accurate samples with low uncertainty and inaccurate samples with high uncertainty during training. To be specific, we define the RAU loss as:

$$L_{RAU} = \log (1 + \frac{n_{AU}}{n_{AC} + n_{AU}} + \frac{n_{IC}}{n_{IC} + n_{IU}})$$

(10)

When $n_{AU}$ and $n_{IC}$ are optimized close to zero, the RAU loss is close to zero, which means the model is certain about the predictions of accurate samples, while there are no overconfident predictions of the inaccurate samples.

**Experiments**

**Experimental Setup**

**Experimental Datasets** We mainly experiment on three open datasets described below. The download links are displayed in Appendix A.

TNEWS, a Chinese dataset proposed by Xu et al. (2020), has identical essence with intent detection. It includes 53360 samples in 15 categories. The provided test set are without gold labels. So we regard validation set as test set and run experiments played in Appendix A.

HWU64, proposed by Liu et al. (2019) to reflects human-home robot interaction, which owns 15726 samples spanning 64 intents. We use one fold train-test split with 9960 training samples and 1076 testing samples.

BANKING77, proposed by Casanueva et al. (2020), which has 13083 samples, 9002 for training and 3080 for testing. This dataset consists of 77 intents in a single domain of online banking inquiry.
Comparison Methods  We compare with the methods as listed below:

**BERT** (Devlin et al. 2018): represents the pre-trained base BERT model.

**Temperature Scaling (TS)** (Guo et al. 2017): is the classical post-processing method learning a single parameter from the dev dataset to rescale the logit after the model is trained.

**Label Smoothing (LS)** (Müller, Kornblith, and Hinton 2019): smooths some part of the one-hot label’ probability to a weighted mixture probability of the none ground-truth labels, which is set to compare our hyperspherical labels.

**Posterior Calibrated (PosCal)** (Jung et al. 2020): minimizes the difference between the predicted and empirical posterior probabilities, which is a competitive recent research.

**Accuracy Versus Uncertainty Calibration (AVUC)** (Krishnan, Tickoo, and Tickoo 2020): proposes an optimization method that utilizes the relevance of accuracy and uncertainty as an anchor for calibration.

Implementation Details  All experiments are taken on BERT (with or without confidence calibration) unless otherwise specified. We employ Adam (Kingma and Ba 2015) as the optimizer and search learning rate in \{4e-5, 5e-5\} with the training epochs in \{19, 23\} and about 40s per epoch. To make full use of the GPU memory, we set the batch size to 256. The type of GPU is Tesla V100. Besides, the KL loss between predicted probability and empirical probability is added optionally in PosCal, AVUC, and our model. More implementation details are shown in Appendix A.

Confidence Calibration Metric  We follow the previous researches and utilize the expected calibration error (ECE) (Naeini, Cooper, and Hauskrecht 2015), which is a common evaluation metric to calculate the calibration error in confidence calibration. ECE separates the predictions of samples into $M$ bins according to the predicted probability called confidence. Then, accumulating the weighted differences between accuracy and confidence in each bin:

$$ECE = \frac{1}{K} \sum_{i=1}^{K} \sum_{j=1}^{M} \frac{|B_{ij}|}{N} |Acc_{ij} - Con_{ij}|$$  \hspace{1cm} \text{(11)}$$

where $|B_{ij}|$ is the size of bin $j$ in label $i$, $N$ is the number of total prediction samples, $Acc_{ij}$ is the empirical probability and $Con_{ij}$ is the average predicted probability for label $i$ in bin $j$ respectively.

Comparison with State-of-the-arts

Comparison Settings. We reproduce the baselines, and the results are almost equal to the published metrics. Based on that, we conduct extensive experiments on the above three datasets to validate the effectiveness of our model. F1 and ECE are considered as the main evaluation metrics.

Comparison Results. As Table 1 illustrated:

1. Regardless of which dataset we choose, our model achieves the best F1 performance among all the calibration methods. Specially, we obtain a significant reduction in ECE, which drops down 10.50% on average compared with the baseline model. This proves the effectiveness of our model to project label vector onto hyperspherical space uniformly and utilize the rebalanced accuracy-uncertainty loss in confidence calibration.

2. All the calibration methods have limited amelioration in the dataset possessing better performance. It makes sense for the reason that the model is well studied on these datasets. Hence the main distribution of its confidence is as high as the accuracy, like 90+%, which results in more credible predictions. So we majorly analyze the TNEWS dataset in the subsequent experiments.

3. In the case of TNEWS dataset, the F1 gains 1.21% over BERT while the ECE decreases remarkably. Furthermore, more information on the rest datasets can be inquired in Appendix B.

Observing Miscalibration

Comparison Settings. We use the reliability diagrams for observation, which is a visual representation of model calibration (Niculescu-Mizil and Caruana 2005). The average confidence within each bin is defined as “Outputs”, while the absolute difference between confidence and accuracy in each bin is defined as “Gap”. The ECE is proportional to “Gap” in some degree, described in Sec. .

Comparison Results. As Figure 4 depicts, although distinct calibration methods still have a miscalibration phenomenon on TNEWS dataset, BERT with calibration can acquire a lower ECE. Especially, our model decreases ECE prominently and less inclined to make overconfident predictions for samples with higher accuracy compared with the AVUC, which manifests the validity of our RAU loss that rebalances accuracy-uncertainty and optimizes accurate as well as inaccurate samples respectively.

### Table 1: Overall comparison with different calibration methods on three open datasets.

<table>
<thead>
<tr>
<th>Model</th>
<th>TNEWS F1</th>
<th>TNEWS ECE</th>
<th>HWU64 F1</th>
<th>HWU64 ECE</th>
<th>BANKING77 F1</th>
<th>BANKING77 ECE</th>
<th>Average F1</th>
<th>Average ECE</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT (Devlin et al. 2018)</td>
<td>54.81</td>
<td>69.66</td>
<td>91.85</td>
<td>17.18</td>
<td>93.61</td>
<td>11.98</td>
<td>80.09</td>
<td>32.94</td>
</tr>
<tr>
<td>TS (Guo et al. 2017)</td>
<td>54.81</td>
<td>49.88</td>
<td>91.85</td>
<td>15.86</td>
<td>93.61</td>
<td>11.87</td>
<td>80.09</td>
<td>25.87</td>
</tr>
<tr>
<td>LS (Müller, Kornblith, and Hinton 2019)</td>
<td>55.29</td>
<td>53.99</td>
<td>92.06</td>
<td>16.51</td>
<td>93.86</td>
<td>11.40</td>
<td>80.40</td>
<td>27.30</td>
</tr>
<tr>
<td>PosCal (Jung et al. 2020)</td>
<td>54.98</td>
<td>68.05</td>
<td>92.03</td>
<td>16.14</td>
<td>93.66</td>
<td>11.93</td>
<td>80.30</td>
<td>32.05</td>
</tr>
<tr>
<td>AVUC (Krishnan, Tickoo, and Tickoo 2020)</td>
<td>55.41</td>
<td>67.98</td>
<td>92.02</td>
<td>15.71</td>
<td>93.83</td>
<td>11.79</td>
<td>80.42</td>
<td>31.83</td>
</tr>
<tr>
<td>HS-RAU (Ours)</td>
<td><strong>56.02</strong></td>
<td><strong>39.99</strong></td>
<td><strong>92.52</strong></td>
<td><strong>16.12</strong></td>
<td><strong>93.89</strong></td>
<td><strong>11.21</strong></td>
<td><strong>80.81</strong></td>
<td><strong>22.44</strong></td>
</tr>
</tbody>
</table>
Table 2: Ablation study on TNEWS dataset by removing the main components, where “w/o” means without, “HS” represents hyperspherical space calibration, and “RAU” indicates rebalanced accuracy-uncertainty loss.

<table>
<thead>
<tr>
<th>Model</th>
<th>ACC</th>
<th>P</th>
<th>R</th>
<th>F1</th>
<th>ECE</th>
</tr>
</thead>
<tbody>
<tr>
<td>HS-RAU</td>
<td>56.39</td>
<td>56.31</td>
<td>55.82</td>
<td>56.02</td>
<td>39.99</td>
</tr>
<tr>
<td>w/o HS</td>
<td>56.32</td>
<td>55.82</td>
<td>55.26</td>
<td>55.52</td>
<td>67.97</td>
</tr>
<tr>
<td>w/o RAU</td>
<td>55.90</td>
<td>56.06</td>
<td>54.75</td>
<td>55.28</td>
<td>42.53</td>
</tr>
<tr>
<td>w/o Both</td>
<td>56.37</td>
<td>56.13</td>
<td>54.81</td>
<td>55.31</td>
<td>68.50</td>
</tr>
<tr>
<td>RAU⇒AVUC</td>
<td>56.21</td>
<td>55.72</td>
<td>55.45</td>
<td>55.55</td>
<td>41.82</td>
</tr>
</tbody>
</table>

Ablation Study

Comparison Settings. To validate the effectiveness of our model components, we gradually get rid of some components, including hyperspherical space and rebalanced accuracy-uncertainty loss. In practice, if hyperspherical space is not employed in the model, we use the typical one-hot vector in Euclidean space to represent the labels.

Comparison Results. As described in Table 2:

1. Taking out any components of our model results in performance reduction, which certifies the validity of all components.

2. Specifically, replacing hyperspherical space leads to conspicuous performance degradation. It shows that projecting labels onto hyperspherical space to obtain a dense label matrix and setting the calibration scale throughout the process can effectively draw down the ECE.

3. Substituting RAU for AVUC causes a minor decline in F1, while ECE gets 1.83% worse. This proves that RAU, which respectively optimizing accurate and inaccurate samples, is beneficial to improve performance. See more results on other datasets in Appendix C.

Discussion

Effectiveness on Noisy Data

Comparison Settings. Model performance under noisy data is an important indicator to measure robustness, as it’s a frequent phenomenon for data to have noise. We randomly mislabel 5%, 10%, 30%, and 50% part of samples’ labels on TNEWS dataset to simulate the noisy environment.

Comparison Results. The experimental results of Table 3 support the statements as below:

1. The experimental results indicate that our model still obtains significant improvement irrespective of how much noise label is in the dataset.

2. Take TNEWS dataset with 30% error labels as an example. F1 increases 3.27%. In the meantime, ECE decreases by 45.73%. During the experiment, we find that although temperature scaling obtains a comparable ECE, the scale parameter turns extremely large, and the output probability is only distributed into one bin. The reason may be that temperature scaling is not suitable for the situation where training set and other sets are labeled differently, as it learns the scale parameter from the development set after the model is trained on the training set.

3. Above results verify the effectiveness of our model on the noisy data, which projects the label vector uniformly onto the hyperspherical space and makes better use of the dense label representations. Hyperspherical label vector is
Table 3: Performance on TNEWS dataset with noise.

<table>
<thead>
<tr>
<th>Model</th>
<th>5% Noisy Labels</th>
<th>10% Noisy Labels</th>
<th>30% Noisy Labels</th>
<th>50% Noisy Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F1</td>
<td>ECE</td>
<td>F1</td>
<td>ECE</td>
</tr>
<tr>
<td>BERT</td>
<td>53.46</td>
<td>72.19</td>
<td>51.44</td>
<td>72.38</td>
</tr>
<tr>
<td>LS</td>
<td>53.95</td>
<td>46.46</td>
<td>52.59</td>
<td>43.81</td>
</tr>
<tr>
<td>PosCal</td>
<td>54.11</td>
<td>52.77</td>
<td>52.14</td>
<td>61.92</td>
</tr>
<tr>
<td>AVUC</td>
<td>54.04</td>
<td>64.47</td>
<td>51.94</td>
<td>61.78</td>
</tr>
<tr>
<td>HS-RAU</td>
<td>54.43</td>
<td>34.37</td>
<td>53.66</td>
<td>35.48</td>
</tr>
</tbody>
</table>

Table 4: Performance of the low-frequency labels on TNEWS dataset, where L-F<sub>1</sub> means the Lowest Frequency label and “Average” means the average performance of the three lowest frequency labels.

<table>
<thead>
<tr>
<th>Model</th>
<th>L-F&lt;sub&gt;1&lt;/sub&gt;</th>
<th>L-F&lt;sub&gt;2&lt;/sub&gt;</th>
<th>L-F&lt;sub&gt;3&lt;/sub&gt;</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F1</td>
<td>F1</td>
<td>F1</td>
<td>F1</td>
</tr>
<tr>
<td>BERT</td>
<td>40.54</td>
<td>44.50</td>
<td>57.54</td>
<td>47.53</td>
</tr>
<tr>
<td>TS</td>
<td>40.54</td>
<td>44.50</td>
<td>57.54</td>
<td>47.53</td>
</tr>
<tr>
<td>LS</td>
<td>40.45</td>
<td>46.56</td>
<td>59.50</td>
<td>48.84</td>
</tr>
<tr>
<td>PosCal</td>
<td>35.44</td>
<td>46.15</td>
<td>60.00</td>
<td>47.20</td>
</tr>
<tr>
<td>AVUC</td>
<td>43.37</td>
<td>48.31</td>
<td>60.88</td>
<td>50.85</td>
</tr>
<tr>
<td>HS-RAU</td>
<td>54.12</td>
<td>47.22</td>
<td>59.06</td>
<td>53.47</td>
</tr>
</tbody>
</table>

Effectiveness on Low-Frequency Labels

Comparison Settings. Class-imbalanced datasets commonly face the long tail problem. To examine the performance of minority labels, we experiment on the three lowest frequency labels on TNEWS dataset, which contains fifteen classes in total. The sum of fifteen labels’ ECE equals the gross ECE. Besides, L-F<sub>1</sub> means the Lowest Frequency label, L-F<sub>2</sub> means the second lowest, and so on. “Average” means the average performance of the three lowest frequency labels.

Comparison Results. As demonstrated in Table 4, our model reaches the highest average F1 of the low-frequency labels, with a 5.94% absolute increase. Apart from that, it works better on the average ECE too. According to the consequences, we can infer that it’s no picnic to learn sparse label features (like one-hot) with low-frequency samples. In contrast, the label features of our model are dense, as we separate the label vector into the hyperspherical space horizontally and use more portion of the output space. More details on the comparison results are shown in Appendix E.

Effectiveness on Different Encoders

Comparison Settings. Different encoders have distinct output spaces. To assess the performance of different encoders, we also horizontally compare with other encoders such as Albert-tiny (Lan et al. 2020), XLNet (Yang et al. 2019), and BERT-large (Devlin et al. 2018) on TNEWS dataset.

Comparison Results. The consequences of different encoders on TNEWS dataset are shown in Table 5. Though different encoders behave diversely from each other, they all acquire a comparable enhancement with the help of our confidence calibration strategy. Results indicate that projecting the output space onto hyperspherical space by our strategy possesses a certain universality, which is not limited to the BERT model.

Conclusion

In this work, we propose a confidence calibration model for intent detection via Hyperspherical Space and Rebalanced Accuracy-Uncertainty loss (HS-RAU). With the help of projecting label vectors onto hyperspherical space uniformly, we make better use of the dense label representation matrix to mitigate the over-confident predictions as well as the whole portion of output space. Through the rebalanced accuracy-uncertainty loss, we better guide the model to respectively optimize the accurate and inaccurate samples. Experimental results indicate that our model obtains a decent rise over SOTA. Especially, we achieve a significant improvement in the confidence calibration metric (ECE) among the calibration methods.

Acknowledgments

The work is supported by National Natural Science Foundation of China (Grant No.62036001) and PKU-Baidu Fund (No. 2020BD021).
References


Chatterjee, A.; and Sengupta, S. 2020. Intent Mining from past conversations for Conversational Agent. In COLING.


Li, W.; Dasarathy, G.; and Berisha, V. 2020. Regularization via Structural Label Smoothing. In AISTATS.


