SNU_IDS at SemEval-2019 Task 3: Addressing Training-Test Class Distribution Mismatch in Conversational Classification

Sanghwan Bae, Jihun Choi and **Sang-goo Lee**
Department of Computer Science and Engineering
Seoul National University, Seoul, Korea
{sanghwan,jhchoi,sglee}@europa.snu.ac.kr

Abstract

We present several techniques to tackle the mismatch in class distributions between training and test data in the Contextual Emotion Detection task of SemEval 2019, by extending the existing methods for class imbalance problem. Reducing the distance between the distribution of prediction and ground truth, they consistently show positive effects on the performance. Also we propose a novel neural architecture which utilizes representation of overall context as well as of each utterance. The combination of the methods and the models achieved micro F1 score of about 0.766 on the final evaluation.

1 Introduction

A new task whose goal is to predict the emotion of the last speaker given a sequence of text messages has been designed, and coined Contextual Emotion Detection (EmoContext; Chatterjee et al., 2019).

Though predicting the emotion of a single utterance or sentence, *i.e.* emotion detection, is a well-discussed subject in natural language understanding literature, EmoContext has several novel challenges. Firstly, the class distribution of training data is significantly different from that of the test data. Consequently, a model trained on the training data might not perform well on the test data. There have been efforts made to address the problem of learning from training data sets that have imbalanced class distribution, i.e. the class imbalance problem (Chawla et al., 2004; Buda et al., 2018). However, they are not applicable to our case, since the imbalance appears only in the test data while the training data can be viewed to be balanced.

We extend the existing methods of addressing the class imbalance problem to be applicable for more general cases where the distributions of the collected training data differ from those of the real population or the data at test time, under the assumption that the validation set is organized carefully to reflect the practical distribution. From experiments and analyses, we show that the proposed methods reduce the difference between two distributions and as a result improve the performance of the model.

The additional challenge we have to consider arises from the fact that utterances having identical surface form may have different meanings due to sarcasm, irony, or etc.. Thus a model should track the emotional transitions within a dialogue. To grasp the context of utterances, we propose a semi-hierarchical encoder structure. Lastly, the texts contain lots of non-standard words, *e.g.* emoticons and emojis. This makes it difficult to exploit typical pre-trained embeddings such as GloVe (Pennington et al., 2014). Therefore, we adopt pre-trained embeddings which are specialized for handling non-standard words. We show that the proposed model largely outperforms the baseline of task organizers by experiments.

2 Related Work

2.1 Contextual Emotion Detection

EmoContext is a emotion classification data set composed of 3-turn textual conversations. The goal of the data set is to classify the emotion in the last utterance of each example given the context. The label set consists of 4 classes: 'happy', 'sad', 'angry' and 'others'. In the training data set, there are about 50% of 'others' class examples and 50% of emotional (happy, sad, angry) examples, which can be viewed as well-balanced. On the other hand, only 15% of examples in the test and the validation data set are labeled as emotional, reflecting the real-life frequency. For more details, refer to Chatterjee et al. (2019).

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 312–317
Minneapolis, Minnesota, USA, June 6–7, 2019.

2.2 Class Imbalance Problem

When some classes have the significantly higher number of examples than other classes in a training set, the data is said to have an imbalanced class distribution (Buda et al., 2018). This makes it difficult to learn from the data set using machine learning approaches (Batista et al., 2004; Mazurowski et al., 2008), since the learned models can be biased to majority classes easily, which results in poor performance (Wang et al., 2016). We give a brief explanation of methods to solve this problem.

Sampling: This type of methods deals with the problem by manipulating the data itself to make the resulting data distribution balanced. The simplest versions are *random oversampling* and *random undersampling*. The former randomly duplicates examples from the minority classes and the latter randomly removes instances from the majority classes (Mollineda et al., 2007).

Thresholding: This method moves the decision threshold after training phase, changing the output class probabilities. Typically, this can be done by simply dividing the output probability for each class by its estimated prior probability (Richard and Lippmann, 1991; Buda et al., 2018).

Cost-Sensitive Learning: This assigns different misclassification cost for each class and applies the cost in various ways, *e.g.* output of the network, learning rate or loss function. For multi-class classification tasks, the simplest form of the cost-sensitive learning is to introduce weights to the cross entropy loss (Han, 2017; Lin et al., 2018):

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{K} w^c y_i^c \ln p(c|x_i), \qquad (1)$$

where N and K denote the total number of examples and classes respectively, $p(c|x_i)$ the predicted probability of i-th example x_i belonging to class c, y_i^c the ground truth label which is 0 or 1, and w^c a class dependent weighting factor. Recent work suggests to use the inverse ratio of each class, *i.e.* $w^c = \frac{N}{N_c}$, where N_c is the number of examples belonging to class c (Wu et al., 2018; Aurelio et al., 2019).

Ensemble: The term ensemble usually refers to a collection of classifiers that are minor variants of the same classifier to boost the performance. This is also successfully applied to the class imbalance problem (Sun et al., 2007; Seiffert et al., 2010), by replacing the resampling procedure in the bagging algorithm with oversampling or undersampling (Galar et al., 2012).

3 Methods for Mismatch Problem

3.1 Sampling

In our case, it is not possible to make the distinction between majority and minority classes; even if the 'others' class is the most prevalent in the training data, the ratio is less than that of the test data. To address this issue, the random minority oversampling technique should be modified, since it assumes that the class imbalance appears only in the training data set. Accordingly, we apply random oversampling or random undersampling to make the distribution of the training data similar to that of the validation data.

3.2 Thresholding

The basic thresholding method mentioned in §2.2 is not sufficient for our case, since we should additionally *bias* the model output distribution to match the test time distribution, not only correcting the imbalance in the training data. When p_r is a probability of training time and p_s is that of validation time, we multiply the predicted probability by the estimated class ratio from validation set as below:

$$\begin{aligned} y_c(x) = p(c|x) &\approx \frac{\boldsymbol{p_s(c)}}{p_r(c)} \cdot p_r(c|x) \\ &= \frac{\boldsymbol{p_s(c)}}{p_r(c)} \cdot \frac{p_r(c) \cdot p_r(x|c)}{p_r(x)} \\ &= \frac{\boldsymbol{p_s(c)} \cdot p_r(x|c)}{p_r(x)}, \end{aligned} \qquad (2)$$

where $p(c) = \frac{N_c}{N}$.

3.3 Cost-Sensitive Learning

The weighted cross entropy loss, described in Eq. (1), can be used for our case, as long as the weights are carefully chosen. Although the reciprocal of class ratio is helpful for learning balanced prediction (Wu et al., 2018), the target distribution between classes is not uniform for our task. Also, the misclassification cost of 'others' class should be larger than those of emotional classes, because the model tends to predict it less than the actual. Therefore, it is reasonable to modify w^c by multiplying the estimated ratio of each class for test time, *i.e.* $w^c = \frac{N^r}{N_c^r} \cdot \frac{N_c^s}{N^s}$, where N^r and N^s denote

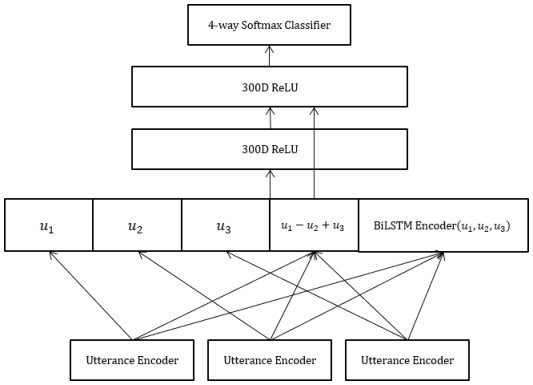

Figure 1: Overall Architecture

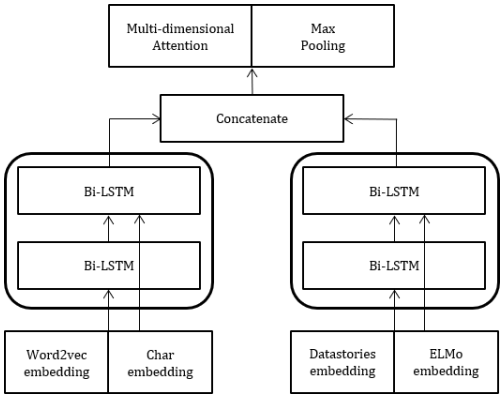

Figure 2: Utterance Encoder

the number of instances in training and validation data set respectively. This corresponds to the term introduced in Eq. (2), as $\frac{N_c^r}{N^r} \cdot \frac{N^s}{N_c^s} = \frac{p_s(c)}{p_r(c)}$. The difference between them is that thresholding utilizes the term in inference phase after training is finished, while weighted cross entropy loss includes it from the training time.

3.4 Ensemble

We combined bagging-based ensemble with sampling techniques represented in §3.1. In addition, we compared ensembles of randomly initialized classifiers using other methods (*i.e.* thresholding and cost-sensitive learning).

4 Model and Training Details[1]

4.1 Overall Architecture

We propose a semi-hierarchical structure to capture the context as well as the meaning of each single utterance. Fig. 1 depicts the overall architecture. Each single utterance representation u_m is encoded by Utterance Encoder described in Fig. 2. In addition, we introduce another bi-directional LSTM (BiLSTM) encoder for higher level representation which receives the outputs of utterance encoder as its inputs. The representation of all context is generated by the concatenation of u_1, u_2, u_3, $u_1 - u_2 + u_3$ and the output of this encoder. Then it is fed to the two 300-dimensional (300D) hidden layers with $ReLU$ activation with shortcut connections and a $softmax$ output layer.

4.2 Utterance Encoder

The proposed utterance encoder has two types of shortcut-stacked bi-directional long short-term memory (BiLSTM) encoder (Nie and Bansal, 2017) to fully exploit four types of lexicon level representations. The first encoder utilizes pre-trained word2vec (Mikolov et al., 2013) embeddings concatenated with trainable embeddings from a character level convolutional neural network (CNN). We also added emoji2vec (Eisner et al., 2016) embeddings to word2vec to map emoji symbols to the same space. The other encoder receives concatenated representations of pre-trained datastories (Baziotis et al., 2017) embeddings and contextualized represenations from ELMo (Peters et al., 2018).

The results from the two stacked BiLSTM encoder are concatenated. We use the multi-dimensional source2token self-attention of Shen et al. (2018) and max pooling to integrate contextualized word level representations to a single utterance representation, as in Im and Cho (2017).

4.3 Training Details

We use 300D Google News word2vec[2] embeddings and 300D pre-trained emoji2vec.[3] Datastories vectors[4] which were pre-trained on a big collection of Twitter messages using GloVe are also 300D. The dimension of character embeddings is fixed to 15, and it is fed to a CNN where the filter sizes are 1, 3, and 5 and the number of feature map for each is 100, thus a 300D vector is generated for each word as a result. To guarantee the same size for ELMo embeddings, a 300D position-wise feed-forward network is applied above them. The hidden states of all the BiLSTMs for each direc-

[1]The implementation of our model is available at https://github.com/baaesh/semeval19_task3

[2]https://code.google.com/archive/p/word2vec/

[3]https://github.com/uclmr/emoji2vec

[4]https://github.com/cbaziotis/datastories-semeval2017-task4

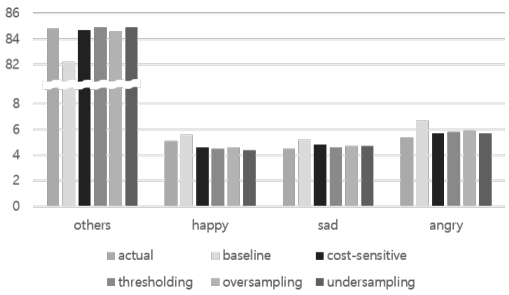

Figure 3: Class distributions on test data. Actual is from the ground truth labels and the others are from the predicted labels by each model. These are the averages of 10 results with random initialization.

tion are 150D and the number of layers is 2.

Our model is trained using the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.001 and a batch size 64. We clip the norm of gradients to make it smaller than 3. Dropout (Srivastava et al., 2014) technique is applied to word embeddings with $p = 0.1$. We chose the best model based on a micro F1 score on the validation set.

5 Experiments

5.1 Effects of mismatch in class distributions

Fig. 3 shows that the class distribution of predictions from the baseline model without any methods applied is substantially different from that of the actual test data. On the other hand, when the proposed methods are applied, the gap becomes much smaller. From this result, we conjecture that the difference in output distributions could be a reason for poor performance of the baseline compared to the proposed methods, as presented in Table 1.

5.2 Single model methods

Table 1 shows the accuracy and micro F1 scores of variants of our methods and baselines on the test set. We report the mean and standard deviation of 10 experimental runs (with the same hyperparameters) for each methods. And all the models were chosen based on their performance on the validation set.

The reported results show that proposed methods except undersampling are effective for enhancing both accuracy and F1 score. This means that alleviating the difference of class distribution is the key factor for the higher performance. In the case of undersampling, since the total size of training data decreases, the model seems to fail to capture the general semantics. The result is consistent with Buda et al. (2018), where the undersampling solely does not bring a performance gain for deep learning models. On the other hand, in our experiments, thresholding and cost-sensitive learning were the most effective approaches when a single model is used.

Approach	Acc	(±)	F1	(±)
Baseline (organizers)	-	-	.587	-
Baseline (ours)	.914	.005	.726	.008
Oversampling	.922	.004	.733	.012
Undersampling	.919	.006	.719	.013
Thresholding	**.924**	.002	.738	.010
Cost-Sensitive	.924	.004	**.739**	.010

Table 1: Comparison of single model approaches on the test set.

Approach	Acc	F1
Baseline (ours)	.921	.743
Oversampling	.930	**.758**
Undersampling	.930	.753
Thresholding	.930	.752
Cost-Sensitive	**.931**	.757
Mixed (submitted)	**.933**	**.766**

Table 2: Comparison of ensemble approaches on the test set. 10 models were used for each ensemble result.

5.3 Ensemble methods

Table 2 reports the comparison of ensemble models. The results show that our methods consistently outperform the baseline. We can see that ensemble with bagging has a great effect on refining class distribution, and in this time, undersampling also showed a good performance. Overall, for ensemble methods, oversampling and cost-sensitive learning performed best. The version we submitted to the leaderboard was the ensemble of different methods selected by their performance on validation set, and achieved the official result of 0.766.

6 Conclusion

In this paper, we proposed several methods for alleviating the problems caused by difference in class distributions between training data and test data. We demonstrated that these methods have positive effects on the result performance. We also presented a novel semi-hierarchical neural architecture that effectively exploits the context and the utterance representation. For future work, we plan to conduct more systematic experiments on other data sets to generalize our results.

References

Yuri Sousa Aurelio, Gustavo Matheus de Almeida, Cristiano Leite de Castro, and Antonio Padua Braga. 2019. Learning from imbalanced data sets with weighted cross-entropy function. *Neural Processing Letters*, pages 1–13.

Gustavo EAPA Batista, Ronaldo C Prati, and Maria Carolina Monard. 2004. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter*, 6(1):20–29.

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754.

Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. 2018. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

NV Chawla, N Japkowicz, and A Kotcz. 2004. Editorial: special issue on learning from imbalanced data sets. sigkdd explor newsl 6: 1–6.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bošnjak, and Sebastian Riedel. 2016. emoji2vec: Learning emoji representations from their description. *arXiv preprint arXiv:1609.08359*.

Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. 2012. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484.

Xiao Han. 2017. Automatic liver lesion segmentation using a deep convolutional neural network method. *arXiv preprint arXiv:1704.07239*.

Jinbae Im and Sungzoon Cho. 2017. Distance-based self-attention network for natural language inference. *arXiv preprint arXiv:1712.02047*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2018. Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence*.

Maciej A Mazurowski, Piotr A Habas, Jacek M Zurada, Joseph Y Lo, Jay A Baker, and Georgia D Tourassi. 2008. Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance. *Neural networks*, 21(2-3):427–436.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

RA Mollineda, R Alejo, and JM Sotoca. 2007. The class imbalance problem in pattern classification and learning. In *II Congreso Espanol de Informática (CEDI 2007). ISBN*, pages 978–84.

Yixin Nie and Mohit Bansal. 2017. Shortcut-stacked sentence encoders for multi-domain inference. *arXiv preprint arXiv:1708.02312*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Michael D Richard and Richard P Lippmann. 1991. Neural network classifiers estimate bayesian a posteriori probabilities. *Neural computation*, 3(4):461–483.

Chris Seiffert, Taghi M Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. 2010. Rusboost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 40(1):185–197.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Yanmin Sun, Mohamed S Kamel, Andrew KC Wong, and Yang Wang. 2007. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378.

Shoujin Wang, Wei Liu, Jia Wu, Longbing Cao, Qinxue Meng, and Paul J Kennedy. 2016. Training deep neural networks on imbalanced data sets. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 4368–4374. IEEE.

Zhenyu Wu, Yang Guo, Wenfang Lin, Shuyang Yu, and Yang Ji. 2018. A weighted deep representation learning model for imbalanced fault diagnosis in cyber-physical systems. *Sensors*, 18(4):1096.