Happy is Farther From Sad Than Joyful: Label-Aware Contrastive Loss for Fine-grained Text Classification

Seyyed Mohammad Reza Modarres
Tehran Institute for Advanced Studies
smrmodares@gmail.com

Mohammad Amin Raeisi
Tehran Institute for Advanced Studies
mhmdaminraeisi@gmail.com

Abstract

Fine-grained text classification involves datasets with many classes with slight differences. The key to elevating performance is guiding the model to differentiate these commonly confusable classes. In this work, we first used the contrastive loss to fine-tune pre-trained language models. Then the label-aware contrastive loss is used to adaptively embed the class relationship into a contrastive objective function to help differently weight classes. For instance, happy and joyful classes must be closer and farther away from sad ones. We experimented with three fine-grained text classification tasks, emotion classification, text classification, and sentiment analysis.

1 Introduction

Fine-grained classification involves distinguishing between classes that have subtle differences among them. For example, we can classify dogs from non-dogs or try a more fine-grained classification of dog species in image classification. In NLP, we have the same thing. For example, in emotion classification, we could try classifying with 27 or 32 labels instead of 4 or 6. Similarly, in sentiment analysis, we could have "very positive" and "positive" instead of only the "positive" class. This involves distinguishing between some closely confusable pairs of classes, such as "happy" and "joyful". This task is challenging due to class interference.

The standard method today is using a pre-trained model like BERT and cross-entropy. However, the problem with this method is that it treats classes the same way, so misclassifying "positive" as "very positive" is not different from "very negative", or "joyful" and "happy" are entirely different from "sad". To improve performance, we can modify the loss to reflect the contrast between different pairs of examples. Here, we differentiate between classes with label-aware contrastive loss similar to Suresh and Ong (2021).

In this work, we implemented the code for supervised contrastive loss as well as the code for label-aware contrastive loss. Then we trained and tested BERT and ELECTRA models with those losses on datasets of the original paper as well as three new datasets. We also found some minor errors in the label-aware contrastive loss formula and corrected them.

2 Related Work

2.1 Fine-grained classification

Fine-grained classification tasks involve finding subtle differences to distinguish between close classes. This problem is challenging because the classes are semantically similar, which makes it difficult for the model to learn the labels (Collins et al., 2018). Recent models have applied state-of-the-art attention mechanisms and multi-task learning to solve fine-grained sentiment classification. For example, Tian et al. (2020) modified the pre-training objectives of language models to include more sentiment-specific tasks, such as sentiment word masking and sentiment word prediction, showed improved performance in fine-grained sentiment analysis. These methods mainly focus on improving the pre-training of language models; here, we focus on improving contrastive fine-tuning to solve fine-grained text classification.

One important fine-grained classification task is emotion recognition. Traditionally, emotion recognition datasets have a small number of emotions (e.g., 4-7). One recent dataset proposed for this issue: Rashkin et al. (2019) introduced Empathetic Dialogues, which contain text conversations labeled with 32 emotion labels.

2.2 Contrastive Learning

Contrastive learning enhances the model's ability to distinguish between different classes. It has a broad usage in computer vision, particularly in self-supervised environments, where such learning guides the model based on similarities between the latent representation of the samples. (Gao et al., 2021) used a contrastive objective to fine-tune pre-trained language models to obtain sentence embeddings and achieved state-of-the-art performance in sentence similarity tasks. Recently (Suresh and Ong, 2021) used a label-aware contrastive loss to improve the fine-tuning objective of a pre-trained language model for downstream tasks involving fine-grained classes.

3 Proposed Method

3.1 RNN and GRU

First, we start our work using the RNN model. To make it work, we preprocessed the input string by doing operations such as lower text, removing stop words, filters using regular expressions, Etc. Then gave processed input to a custom tokenizer(code from Madewithml). We then used GRU in the same manner.

3.2 BERT and ELECTRA Base Model

The RNN and GRU models are not the best choice; no surprise there. So we fined-tuned pre-trained language models like BERT. We implemented cross-entropy as well in order to get ready to implement a custom loss.

3.3 Supervised Contrastive Learning

Having the base model, we now turn to contrastive learning. We want to change the loss to guide the model to differentiate different classes in a better way. We implemented the model using Hugging Face and wrote our custom loss function. Our supervised contrastive loss (SCL) is similar to Gao et al. (2021).

$$\sum_{i=1}^{K} -\frac{1}{|P|} \sum_{p \in P} \log \frac{exp(h_i.h_p/\tau)}{\sum_{k \in I \setminus i} exp(h_i.h_k/\tau)}$$

In the above formula, K is the number of data points, P is the positive class(i.e., instances with the same class as instance i), h_i is a representation of the sentence, τ is a hyper-parameter to scale the dot product, and
$I = \{1, ..., K\}$.

In this part, we mainly used the Hugging Face library. In the first step, we used AutoModelForSequenceClassification. However, the output accuracy was no better than random! The problem occurred because the model had its classifier, and we had logits in the loss function, but we needed a representation of the input, not the logits. We needed a vector to represent a sentence, but we had a bunch of vectors with 768 elements. Our first attempt was to flatten the vectors to make a very long vector representation, but we faced a memory error. That was because the resulting vector was too large. Then we tried another way, dropping [CLS] and [SEP] tokens and taking the mean of all other ones to have one vector with 768 elements. At last, we just used the [CLS] token as representation. In our experiments, these two last methods had almost the same results, therefor we used [CLS] because it needed less computation than taking the mean. To implement that, we had the model fine-tuned for the task with SCL, and after fine-tuning, we trained a classifier with cross-entropy loss, to do feature extraction. We also implemented another version ourselves in which we do not only use L_{SCL} but a linear combination of cross-entropy loss and L_{SCL}.

Notice: The code was our implementation and different from the code of the mentioned paper.

3.4 Label-aware Contrastive Loss

Now we want a loss function to understand the relation between classes adaptively. For example, misclassifying positive as very positive is different than very negative. The loss function is like what Suresh and Ong (2021) proposed.

We have a model consisting of three components, an encoder to get representations of inputs, a weighting network, and a classifier, each of which gives us a loss , L_e, l_w, L_{LCL} respectively. We optimize the whole model with a linear combination of those losses. (See Figure 1.)

$$L_i = \sum_{p \in P} w_{i,y_i} \log \frac{exp(h_i.h_p/\tau)}{\sum_{k \in I \setminus i} w_{i,y_k} exp(h_i.h_k/\tau)}$$

L_{LCL} is very similar to L_{SCL} with only a weight factor that differs. w_{i,y_i} says the relationship between an input x_i and a label y_k. Here $w \in R^C$ where C is the total number of classes and is given by the formula:

$$w_i = \frac{exp(h_i)}{\sum_{c=1}^{C} exp(h_c)}$$

Here, $w_i = \{w_{i,c}\}_{c=1}^{C}$. In fact, w is the weighting model's logits output. Now we can compute L_{LCL}.

$$L_{LCL} = \sum_{i=1}^{K} -\frac{1}{|P|} L_i$$

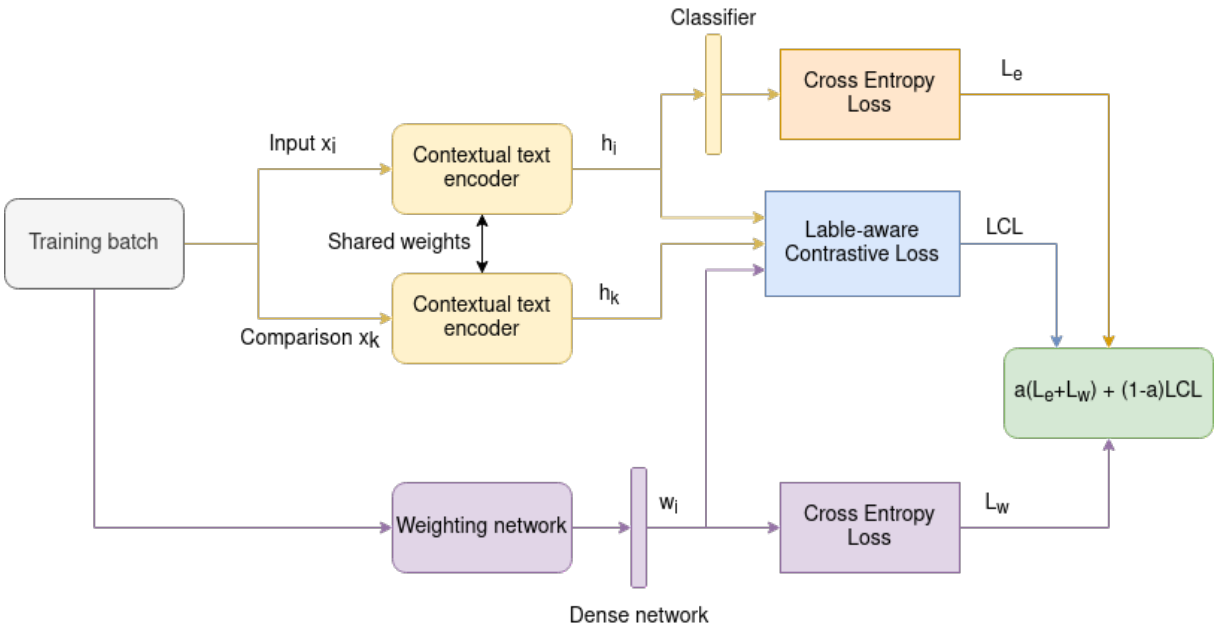

Figure 1: Shows the illustration of the training strategy used in our Label-aware Contrastive Loss approach. We compare every instance with every other instance in the batch. Encoder, weighting network, and classifier optimize simultaneously.

Notice that the above formulas are slightly different from the cited paper, there were some typos there, and we have corrected them. To implement this model, we defined a new pytorch model. In the forward method, we defined all three components (similar to Figure1.) and returned a linear combination of their losses as L_f to optimize the model.

$$L_f = \alpha(L_e + L_w) + (1 - \alpha)(L_{LCL})$$

Here, α is a tunable loss scaling factor determining the effect of different losses. Parameter α cannot be too close to one because it makes the effect of L_{LCL} insignificant.

We observed that the encoder, the weighting network, and the classifier are optimized during the training phase. Then we re-implement SCL in the same way in the forward function. So encoder and classifier would optimize together rather than separate phases of fine-tuning and feature extraction.

4 Results

We evaluate our models with costume losses using three tasks, Emotion Classification, text classification, and Sentiment Analysis. To do these experiments, we load datasets from Hugging Face. For emotion classification, we used the following datasets:

- Empathetic Dialogues(ED): It is a dataset of two-way conversations between a speaker and a listener. We only used the first turn, the speaker expressing an emotional happening. So we trained on the prompt(string) and context(string) columns. The context column consists of 32 emotions which we had to map into [0,31].

- GoEmotions: A dataset of Reddit comments labeled with 27 emotions. Some samples have multiple labels we only used the first one.

- EmoInt: It consists of tweets labeled with one of 4 emotion categories.

For Sentiment Analysis:

- SST-5: It consists of movie reviews annotated for the sentiment task with five classes (very positive, positive, neutral, negative, very negative).

- SST-2: It consists of movie reviews annotated for the sentiment task. This one is a binary classification.

- Twitter-sentiment-analysis: Includes tweets with negative/positive classes.

- Parsinlu_sentiment: A dataset in Persian that contains reviews about foods and movies. It contains labels from -3 to +3. The training

Dataset	Empathetic Dialogues	GoEmotions	EmoInt	SST-5	SST-2
Number of Classes	32	27	4	5	2
$\text{BERT}_{\text{base}}$	55	60.2	**85**	52.5	93.5
$\text{ELECTRA}_{\text{base}}$ + Cross-Entropy Loss	56.4			55.6	94.5
$\text{ELECTRA}_{\text{base}}$ + SCL	56.8	56.8	33.1	**58.3**	**94.8**
$\text{ELECTRA}_{\text{base}}$ + LCL	**57.7**	**64.1**	34.2	57.1	94.8

Table 1: Best test results for emotion classification and sentiment analysis tasks. We compare the results of an ELECTRA encoder trained with cross-entropy loss, Supervised Contrastive Loss (SCL), and Label-aware Contrastive Loss (LCL).

was done on the whole dataset but tested only on the movies portion of the test set. We used mBERT as a language model for this dataset. But the results of mBERT with cross-entropy were almost the same as mBERT with custom loss.

Finally, for the text classification:

- Ag-news This dataset is a collection of more than one million news articles and has four classes: Business, Sci/Tech, Sports, and World.

One thing to notice is that even though the number of labels in ED is much more than SST-5, they get very close accuracy in different models.

We ran multiple experiments on these data sets, and our best test results are shown in Tables 1 and 2. Also, we trained GRU and RNN models on ED dataset for comparison purposes. The accuracy for the RNN model was 25%, and for GRU, 30%.

It seems that ELECTRA does the classification task better than BERT. We got 55% accuracy for BERT and 56.4% for ELECTRA on ED. Also, on SST-5, we got 50% for BERT-cased, 52.5% for BERT-uncased, and 55% for ELECTRA. To get these base results, the model was fine-tuned for ten epochs with a batch size of 8.

The LCL is very sensitive to its hyper-parameters and settings; therefore, by choosing poorly, one can end up with terrible results. We did a systematic search on SST-5 to find a proper setting. We choose α form $\{0.1, 0.2, 0.7\}$, and τ from $\{0.07, 7, 10, 100, 1000\}$, and batch size from $\{4, 8, 32\}$. We found best setting to be $\alpha = 0.2$, $\tau = 10$, batch-size = 32.

We tried another loss function similar to SCL as an experiment, but the amount of t was not fixed this time. It changes concerning the size of the positive set. It turned out to be a disaster because it made the embedding space random, and indeed we got random results.

Dataset	AG News	TSA	PNS
$\text{BERT}_{\text{base}}$	94.5	84.2	76
$\text{ELECTRA}_{\text{base}}$ + LCL	**94.7**	**85.7**	76

Table 2: This table shows the test results of three datasets we did experiment on in addition to the datasets in Suresh and Ong (2021). Datasets are ag-news, twitter-sentiment-analysis(TSA), and Parsinlu-sentiment(PNS).
Notice: We used mBERT to train the Persian dataset.

An explanation is needed for the results of the EmoInt dataset because both SCL and LCL give us poor results. We observed that the base BERT model starts at 28% accuracy and gradually reaches 85%. However, our model stuck around 33% and did not improve in 10 to 15 epochs. Even though we tested multiple settings, no improvement was achieved.

5 Discussion

Transformer models like ELECTRA are known to be good, as we observed with our experiment against the RNN and GRU models for the classification task. Using transfer learning with pre-trained language models; can save a lot of computation and training time and achieve better results, and it is the standard approach today. In this work, we wanted to change the loss function to guide the model to better differentiate between classes. For this purpose, the usage of SCL is reasonable because it puts instances in the same class closer together and every other instance at a distance. However, all negatives are not equal. Therefore it makes sense to have a weight associated with each class as what LCL does. Though this weighting is adaptive, there is so much a model can learn in a few epochs, which may be why we have yet to get better results with LCL with respect to SCL in some

cases.

Acknowledgments

We appreciate the time and guidance from prof. Pilevar and our mentors M.Zakizadeh and K.Eskandari. This project was done using the hardware provided by the Tehran Institute for Advanced Studies(TeIAS).

References

Edward Collins, Nikolai Rozanov, and Bingbing Zhang. 2018. Evolutionary data measures: Understanding the difficulty of text classification tasks. *Proceedings of the 22nd Conference on Computational Natural Language Learning*, page 380–391. Brussels, Belgium. Association for Computational Linguistics.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. ArXiv preprint arXiv:2104.08821.

Madewithml. https://madewithml.com/courses/foundations/recurrent-neural-networks/. Code for RNN and GRU.

Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. 2019. Towards empathetic opendomain conversation models: A new benchmark and dataset. *In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, page 5370–5381. Florence, Italy.Association for Computational Linguistics.

Varsha Suresh and Desmond C. Ong. 2021. Not all negatives are equal: Label-aware contrastive loss for fine-grained text classification.

Hao Tian, Can Gao, Xinyan Xiao, Hao Liu, Bolei He, Hua Wu, Haifeng Wang, , and Feng Wu. 2020. Skep: Sentiment knowledge enhanced pre-training for sentiment analysis. *In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, page 4067–4076. Online. Association for Computational Linguistics.