Incorporating Lable-Aware Contrastive Loss into BERT-related models for Fine-grained Emotions Classification

Haitao Huang
University of Wisconsin - Madison
haitao.huang@wisc.edu

Roy Wen
University of Wisconsin - Madison
zwen42@wisc.edu

1 Introduction

Over the past decade, we have seen numerous impactful applications in natural language processing. When searching a question on *Google*, users have better experiences thanks to the autocomplete systems. There are also better chatbots that can complete tasks like reserving hotels, flights, etc. More advanced chatbots can even chat with users in different tones (Roller et al. (2020)). Behind the scene are the breakthroughs of information retrieval systems, sentiment analysis, and deep learning techniques.

In this project, we focus on sentiment analysis, a branch of the vast field. Sentiment analysis has been a popular research topic for a long time and has broad applications. It can be used to find quotations, sarcasm to persons, organizations, and ideas related to political discussions (Gamon et al. (2008)). Also, sentiment analysis would be helpful for creating better non-player characters in video games (Bergsma et al. (2020)).

The approaches for solving sentiment analysis tasks have been evolving these years. In early last decade, lexicon-based approaches and traditional machine learning approaches were more commonly used. For instance, Taboada et al. (2011) proposed a lexicon-based method that uses Semantic Orientation Calculator to solve the polarity classification task. Boiy and Moens (2009) compared the performance of SVM, Naive Bayes Model, and Maximum Entropy classifier under a multilingual setting. More recently, many researchers resorted to deep learning models. Wang et al. (2016) introduced an attention-based LSTM model. Dos Santos and Gatti (2014) applied deep convolutional neural networks to solve semantic analysis tasks of short texts.

We also observe that the scope of sentiment analysis is getting wider. From the dataset that features affective texts in news headlines (Strapparava and Mihalcea (2007)), to the one that introduces and expands *Twitter* tasks (Rosenthal et al. (2014)), researchers turn to study fine-grained emotions and build their datasets from more diverse domains.

For this project, we analyze the fine-grained emotion classification task using the GoEmotions dataset, compiled by Demszky et al. (2020). We need to classify *reddit* comments into twenty-eight classes and each comment could be multi-labeled. The baseline model is obtained by fine-tuning BERT for the downstream task. In our work, we first implement some classic models such as Character-level CNN and LSTM to solve the task. During the training of those models, we observe that the naive binary cross-entropy may not be perfectly suitable for this problem. We discuss this finding in section 5. To deal with this issue, we adopt an innovative loss function which is called Label-aware Contrastive Loss. We also examine the possibility of incorporating character-level information when doing classification. The contributions of our paper are summarized as follows:

- We implement Character-level Convolutional Neural Networks and bi-LSTMs as a preliminary exploration. Their training behaviors suggest that using a plain binary cross-entropy loss function is inadequate.

- The traditional Binary Entropy Loss is often treated as a bias loss, since it treats all labels are totally irrelevant. In our paper, we used an innovative loss called Lable-aware Contrastive Loss (LCL) to improve the model performance. The original version is based on single outputs, we adjust it for a multioutput case.

- We incorporate a strong classification ability model (ELECTRA) to capture the relation among labels.

- We incorporate character-level information by applying CharBERT.

2 Relevant Works

2.1 Character-level Convolutional Neural Networks (CharCNN)

In this work, the authors examine the use of character-level convolutional neural networks for text classification (Zhang et al. (2015)). The authors treat text as a kind of raw signal at the character level. The key components they have built are a 1-D convolution module and the temporal or 1-D max-pooling layer. Their empirical results show that the pooling layer is essential for training the model.

They define a discrete input function g and a kernel function f. The convolution h is conducted on g and f. To be more specific, $g(x) \in [1, l] \rightarrow R$, $f(x) \in [1, k] \rightarrow R$, the convolution function $h(y) \in [1, \lfloor (l-k)/d \rfloor + 1] \rightarrow R$ with stride d is defined as

$$h(y) = \sum_{x=1}^{k} f(x) g(y \cdot d - x + c)$$

Where $c = k - d + 1$ is offset constant.

The authors also propose a way to encode characters in a text. They only consider a finite set of characters with length m. English alphabets, 10 digits, and 33 other characters are included. Then, each character is represented in a 'one-hot' encoding manner. Characters not in the set would be defined as vectors with all zeros.

2.2 Bidirectional LSTM

Long Short-term Memory, or LSTM, is a mechanism that is proposed by (Hochreiter and Schmidhuber (1997)) to cope with the gradient vanishing problem of vanilla recurrent neural networks. The authors of LSTM introduce a component named cell state and the corresponding controlling unit named gate.

In LSTM, there are three types of gates: forget gate f, input gate i, and output gate o. The three gates control how much information to pass forward at each step utilizing the sigmoid function. The input of each gate is the same: hidden state h and input sequence x. For each gate and cell state, there are weight matrix W and bias terms b that transform the inputs.

Suppose at time step t, we have the input x_t, previous hidden state h_{t-1}, previous cell state C_{t-1}. The important operations are:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) : forget\ gate$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) : input\ gate$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) : output\ gate$$

$$\hat{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C) : raw\ cell$$

$$C_t = f_t \cdot C_t + i_t \cdot \hat{C}_t : updated\ cell$$

$$h_t = o_t \cdot \tanh(C_t) : updated\ state$$

With a simple extension of training the model with sequences in both positive and negative time directions Schuster and Paliwal (1997), we obtain the bi-directional LSTM (biLSTM).

2.3 BERTweet

BERTweet is a pre-trained language model for English Tweets proposed by Nguyen et al. (2020). It has the same architecture as $\text{BERT}_{\text{base}}$ (Devlin et al. (2018)), but it's trained using the RoBERTa pre-training procedure (Liu et al. (2019)). They used an 80GB pre-training dataset of uncompressed texts, containing 850M Tweets. They also converted emojis to respective strings in their tokenization process, and applied normalization strategies on translating word tokens of user mentions and web URLs into special tokens.

As a result, they outperformed many downstream tasks including POS tagging, NER, and text classification when the number of classes is small. The BERTweet model beats the original $\text{RoBERTa}_{\text{base}}$ model by advancing 2 % on average for those tweet-related tasks. They provide two pre-trained models in the `Transformers` library (Wolf et al. (2019)). The `bertweet_base` is the model they mentioned above (with 768 hidden states.). The `bertweet_large` is based on $\text{RoBERTa}_{\text{large}}$ model (with 1024 hidden states.).

2.4 Label-aware Contrastive Loss

The Label-aware Contrastive Loss (LCL) was proposed by Suresh and Ong (2021). They adapted from self-supervised (Chen et al. (2020)) and supervised (Khosla et al. (2020)) version of Contrastive Loss (SCL). They kept almost the same setting as the supervised version of Contrastive Loss but added a weighting vector $\boldsymbol{w}_i \in \mathbb{R}^C$, where C is the total number of classes for each entry i. In terms of pret-rained model, they chose $\text{ELECTRA}_{\text{base}}$

(Clark et al. (2020)), due to its strong performance on downstream task like text classification. The above induction is mainly for the single output scenario. However, in our dataset, we need to apply this kind of contrastive loss in the multioutput scenario. See section 3.1 for more details.

The main purpose of proposing this kind of method is to fully utilize the "similarity" information. Especially in fine-grained emotion methods, the labels themselves are not totally independent. For example, "sad" and "devastated" should be semantically closer than "sad" and "happy". So, the contrastive loss can help the model differentiate these insight relationship more rather than treating them all independent.

The results of LCL outperformed many emotion classification tasks by advancing approximately 1% on average in accuracy compared with SCL (Khosla et al. (2020)) model as they proposed. And it has approximately 3% advance compared with original BERT model.

3 Methods

3.1 Contrastive Loss in Multi-outputs case

A potential work for us is to incorporate contrastive loss, especially LCL as we mentioned in section 2.4. The fined-grain emotion is a complicated classification problem. And it's so subjective that different people have different attitudes towards sentences. This kind of fact tells us the emotional labels themselves have strong connections among them, making it difficult to distinguish them. In the Binary Cross-Entropy Loss setting, we treat all labels the same which could cause the bias on learning the features. A natural idea to solve them is applying contrastive loss, which is also called contrastive learning. The goal is to let the similar labels are "closer" in feature space spanned by sigmoid output vectors. We proposed a multi-outputs contrastive loss rather than the original single-output loss in the following paragraph.

Figure 1 is an illustration for multi-output LCL in feature space. It minimizes the distance between positive samples while maximizing the distance between negative. It pushes the more negative samples further away from positive samples, but gives a tolerance for some samples with partial same emotions.

Figure 2 is an illustration of training strategy for our model structure under LCL setting. The two contextual encoder $\mathbf{\Phi}$ shares the same weight. We

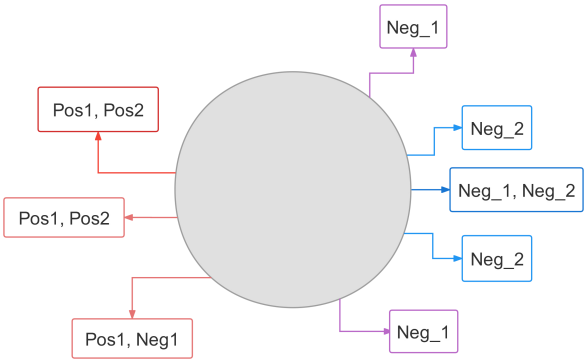

Figure 1: Feature space in Multi-output LCL setting

follows the basic idea of contrastive learning. In the contextual encoder, every sample from the training batch is compared against every other sample in the LCL function. At a testing time, only the contextual encoder is used. The weighting encoder provides us with a weighting vector that adjust the weight of contrastive loss in LCL setting.

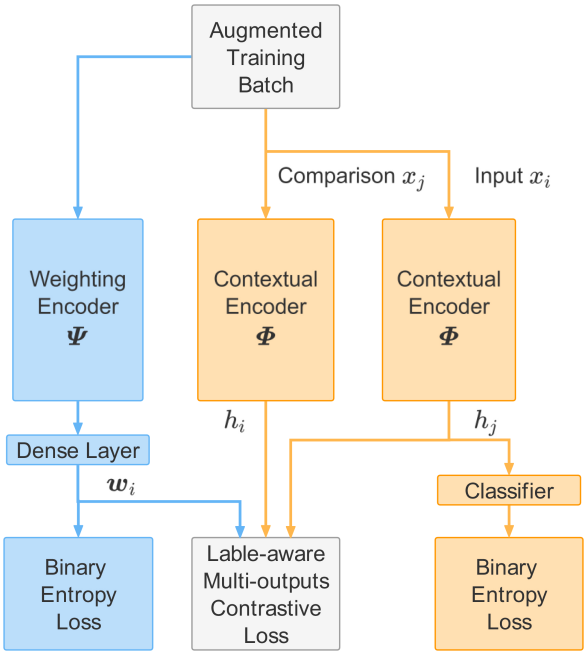

Figure 2: Model Structure with LCL. The two contextual Encoders share the same weights. $\boldsymbol{w}_i$ is the weight for adjusting contrastive loss in LCL. h_i is a normalized representation vector of x_i from contextual encoder.

A detailed picture is as follows. Denote some sets of positives as $\mathcal{P}_k$ (k depends on data x_i) and a set of negatives as $\mathcal{N}$. A batch of sample with sample size K. Then apply augmentation to all K samples to produce K augmented data points. Denote $\mathcal{I} = \{1, 2, \cdots, 2K\}$. That is, each sample will appear twice in the $\mathcal{I}$. For a given data x_i, denote the true labels for x_i are $Z_i = \{z_{i,1}, z_{i,2}, \ldots z_{i,k}\}$. We

include the samples belonging to the class $z_{i,k}$ in its positive set $\mathcal{P}_{z_{i,k}}$ with size $|\mathcal{P}_{z_{i,k}}|$. The mathematical form is $\mathcal{P}_{z_{i,k}} = \{x_j : i \neq j \text{ and } z_{i,k} \in Z_j\}$. And we denote $\mathcal{P} = \cup_k \mathcal{P}_{z_i,k}$. Denote h_i is the normalized (for similarity comparison) representation vector of x_i obtained from a contextual encoder Φ. And a weighting vector $\boldsymbol{w}_i \in \mathbb{R}^C$ which we just mentioned in the previous section (Section 2.4). C is just the number of labels. Then the adapted similarity function for each entry i would be,

$$\mathcal{L}_i = \sum_{p \in \mathcal{P}} \left(\sum_k w_{i,z_{i,k}} \right) \times \log \frac{\exp(h_i \cdot h_p / \tau)}{\sum_{j \in \mathcal{I} \setminus i} \left(\sum_k w_{i,z_{j,k}} \right) \exp(h_i \cdot h_j / \tau)} \quad (1)$$

Where τ is the temperature hyper-parameter. $w_{i,z_{j,k}}$ indicates the relationship between an input x_i and label $z_{j,k}$. And finally, the loss function for each training batch would look like,

$$L_{LCL} = \frac{1}{2K} \sum_{i=1}^{2K} -\frac{1}{|\mathcal{P}|} \mathcal{L}_i \quad (2)$$

Besides the LCL, we also calculate the classification loss for our contextual encoder $\boldsymbol{\Phi}$, which is calculated through Binary Cross-Entropy loss, denoted as $L_{\boldsymbol{\Phi},BC}$. By far, we reached the loss. Can we back propagate the gradient and update the parameters? No, we actually didn't have a well-defined weights for $\boldsymbol{w}_i$. To address this, we could define an another weighting neural network $\boldsymbol{\Psi}$. The input batch is fed into $\boldsymbol{\Psi}$ and output is optimised using Binary Cross-Entropy loss, which is denoted as $L_{\boldsymbol{\Psi},BC}$. Then, weights are obtained from the softmax layer,

$$\boldsymbol{w}_i = \frac{\exp(h_i^{\boldsymbol{\Psi}})}{\sum_{c=1}^{C} \exp(h_i^{\boldsymbol{\Psi}})} \quad (3)$$

Where $h_i^{\boldsymbol{\Psi}}$ is the output of weighting network $\boldsymbol{\Psi}$ for given entry i.

At last, we can combine all losses together to obtain the final loss function. We combined these three losses together, the final loss is defined as follows,

$$L = \lambda(L_{\boldsymbol{\Phi},BC} + L_{\boldsymbol{\Psi},BC}) + (1 - \lambda) L_{LCL} \quad (4)$$

λ is a hyper-parameter that denotes how many weights we should take into consideration when comparing with direct Binary Cross-Entropy loss and Lable-aware Contrastive Loss.

3.2 ELECTRA

ELECTRA is a pre-training approach that trains two transformer models: the generator and the discriminator, proposed by Clark et al. (2020). They discovered that ELECTRA has better performance than BERT and RoBERTa in many downstream text classification tasks including CoLA, SST, etc. For a downstream task like emotion classification, ELECTRA might be able to obtain a better score than the BERT-based model. We can replace BERT model with ELECTRA directly since they are both transformer-based model, the inputs and outputs of them are similar.

3.3 CharBERT

CharBERT is a character-aware pre-training model proposed by Ma et al. (2020). The authors design a dual-channel architecture to utilize the word-level and character-level information. They introduce two key modules: character encoder and heterogeneous interaction.

For a tokenized text input, the authors first split the tokens into characters and transform them into fixed-size vector. Then, they apply a bidirectional Gated recurrent units layer on the all the characters in that text input, not just within one word. This allows the model to extract both character and word information.

After obtaining the character embeddings and the original word tokens, the authors feed them into the same transformer layer. After each transformer layer, they apply heterogeneous interaction on the character features and token features. The two features are first fused by a convolutional layer then split by a fully connected layer with GeLU activation function. The authors suggest that this procedure let the two features enrich one another.

CharBERT has achieved satisfactory results on question answering and sequence labeling tasks. But its performance on sentiment analysis is yet to explore. We know that BERT is good at extracting contextual information and often gives satisfactory results. But the text data in GoEmotions, built on *reddit* comments, are mostly short texts and may include emojis and emoticons. By default, the BERT tokenizer does not consider those emoji or emoticons. Though technically these characters or symbols are not text information, the affective meanings behind a sentence could vary significantly with or without them. Also, observing that using CharCNN (focus on characters within a word) is not

enough for our task, we would like to try CharBERT.

4 GoEmotions Datasets

GoEmotions dataset was proposed by Demszky et al. (2020). The first dataset that is built on Reddit comments for up to 28 emotions (27 emotions + neutral) classification. All samples are manually selected and annotated. When making selections, the authors removed the subreddits that are not safe for work, reviewed identity comments, and removed offensive samples towards a particular ethnicity, gender, sexual orientation, etc. They also reduce some of the subreddits with little representation of positive, negative ambiguous, or neural sentiment for sentiment balancing. At the last stage of selecting examples, they assigned Mask using a BERT-based Named Entity Tagger. For example, they masked proper names referring to people with a [NAME] token. In the annotation aspect, the number of raters per example would be 3 or 5. However, even humans have disagreement on emotion classifications. As reported in their paper, only 94% of examples reach 2+ raters agreeing on at least 1 label, and only 31% examples with 3+ raters agreeing on at least 1 label. They marked some samples as unclear emotions when the annotators have a large disagreement on which emotion labels to classify. In our experiments, we manually deleted the samples which are labeled as unclear to classify.

In the paper, they also proposed a hierarchical grouping for their emotions. A sentiment level divides the labels into 4 categories - *positive*, *negative*, *ambiguous* and Neutral. In the following study, we take this sentiment level as a reference for ordering the emotions in t-SNE plot.

Currently, there are two different versions of GoEmotions. One is the dataset on which they implemented their models and reported their results, we called it `GoEmotions_base`. The other is the raw dataset without specific selection, we called it `GoEmotions_raw`. The total effective sample size (deleted the samples with unclear emotion labels) of `GoEmotions_base` and `GoEmotions_raw` dataset is 54263, 207214 separately. Since the `GoEmotions_raw` wasn't processed with fine-grained selection process, we expected to get a lower accuracy result on it. In our experiments, we split both datasets into training, validation and test dataset separately. The effective sample sizes in each dataset are present in Table 1.

	Base	**Raw**
Effective samples	54263	207214
Training samples	43410	166251
Validation samples	5426	20782
Test samples	5427	20781

Table 1: GoEmotions Datasets size

Below implementations are all done by using `GoEmotions_base` dataset[1].

5 Experiments

5.1 biLSTM

5.1.1 Model Architecture

Before we implement BERT, as an exploration, we examine the performance of biLSTM models. We use pre-trained word embeddings from GLoVe. To be more specific, the embedding dimension is 200 and the pre-training data is a twitter dataset with 27 billion tokens. We use two biLSTM layers with hidden dimension 256. Between the fully connected layers, we add a dropout of probability 0.7, which corresponds to the setting in the GoEmotions paper.

5.1.2 Experiment Dataset and Settings

We train the biLSTM model on the original version of GoEmotions. The optimizer is Adam with a learning rate of 1e-3. We adopt two types of training schedule. In the first schedule, we terminate the training process when the validation loss does not go down for 3 epochs. In the second schedule, we train the model for a fix number of epochs. We have some interesting findings from this two experiments, which leads to our thoughts on using a different loss function.

5.1.3 Results and Findings

We use F1-score as the evaluation metric. In our experiment, with a proper hyperparameter setting, the biLSTM model has a macro-averaged F1-score of 0.42, which is better than the biLSTM model in the original paper. The detailed F1-scores for the 28 emotions are listed in the appendix.

When training our model, we observe that the model obtained by early stopping (Denoted as

[1]We actually tried several models on GoEmotions_raw, but the result is significantly lower than GoEmotions_base. (Dropped from 0.46 to 0.27 in macro F1 score for $\text{BERT}_{\text{base}}$ model)

biLSTM$_{es}$) fails to recognize almost half of the 28 emotions. Though for some emotions, biLSTM$_{es}$ achieves even better F1-score than the base BERT, its overall performance is poor.

For the model with fixed epochs (denoted as biLSTM$_{fix}$), we find that biLSTM$_{fix}$ leads to lower F1-scores of well-classified emotions in biLSTM$_{es}$. But biLSTM$_{fix}$ has a much better overall performance. In the figure below, we show three emotions with the highest F1-scores on biLSTM$_{es}$, three emotions with the lowest F1-scores, and their F1-scores on biLSTM$_{fix}$.

Comparison of F1 Scores		
Emotions	biLSTM$_{es}$	biLSTM$_{fix}$
gratitude	0.91	0.91
amusement	0.75	**0.76**
love	**0.74**	0.72
grief	0.00	**0.36**
pride	0.00	**0.36**
relief	0.00	**0.13**
neutral	**0.63**	0.56

Table 2: Selected Results under Two Training Schedules

One explanation could be that the GoEmotions dataset is unbalanced. The biLSTM model tends to focus more on the dominant classes in the datasets. In our case, the emotion class *neutral* takes more than 33% of the training data. Thus, ignoring some classes with much fewer samples might not increase the binary cross-entropy loss to a large degree. This finding encourages us to try different loss functions and other model architectures.

5.2 Direct implementation using BERT and BERT-related models

5.2.1 Model Architecture

BERT model is a powerful language representation model and has been used on many NLP tasks. We finetuned a BERT$_{base}$ (Devlin et al. (2018)) in our experiment as our baseline performance. In our implementations, we took the first token ([CLS]) representative vectors as BERT output, and added a linear layer with Tanh activation function and then connected to our output layers (the number of hidden states in our output layers is the number of labels). We also applied a dropout layer to prevent overfitting issues. Set the dropout probability to be 0.3. At last, we applied the sigmoid function on our output vectors for computing Binary Cross-Entropy Loss.

5.2.2 Experiment Dataset and Settings

We did the experiment on the base (initial) version of GoEmotions dataset with 28 labels. Set the training epoch to be 6 and batch size to be 8. Noticed that in previous section 5.1.3, we found that a slightly overfitting is good for our models to classify the low frequency emotions, we used a slightly more epochs for better F1 scores. We used Adamw optimizer with a fixed learning rate 1e-05 and Binary Cross Entrophy loss as training loss criteria. We calculated the results from test dataset after completing hyper parameter tuning on validation dataset.

5.2.3 Results

We set our classification threshold to be 0.5. We implemented the BERT, CharBERT, BERTweet, ELECTRA models directly (See Table 3). We treat the BERT model as our baseline since it obtained a similar result compared with Demszky et al. (2020) result (F1 scores are both around 0.46.). The ELECTRA model obtains the best F1 score (0.479), which is approximately 1.7% increase compared with BERT model. Overall, the ELECTRA model outperformed other models in this direct implementation experiment.

	Macro-Average		
Models	P	R	F1
BERT_base	**0.606**	0.398	0.462
CharBert	0.538	0.427	0.466
BERTweet	0.574	0.423	0.469
ELECTRA	0.584	**0.442**	**0.479**

Table 3: Models Results in test dataset for direct implementations. P and R stand for Precision and Recall separately.

5.3 BERT and BERT-related models in LCL settings

5.3.1 Model Introduction

Lable-aware Contrastive Loss (LCL) is a method to compensate the bias appeared in loss when we are doing classification experiment. The original author Suresh and Ong (2021) only applied LCL in single output setting. Here, we generalize the LCL to multi-output setting to be able to make prediction on GoEmotions multi-output dataset.

Figure 2 is the model structure for models in LCL settings. There are mainly two encoders, one is our main encoder, the contextual encoder Φ. We used this encoder as our classifier to make predictions. The other is a model helper, the weighting encoder Ψ, which is used to calculate the weighting parameters used in LCL.

We also applied the same fully connection layer, dropout layer and activation function settings as we showed in section 5.2.1 for comparison.

5.4 Experiment Settings

We set the training epoch to be 9 and batch size to be 16, which let the model capture more information in weighting in a single training batch. We used Adamw optimizer with a fixed main learning rate 4e-05. The default temperature parameter τ we tried is 0.3, and the default weighting parameter λ is 0.5. We calcuated the results from test dataset after completing hyper parameter tuning on validation dataset.

5.4.1 Overall Results

	Macro-Average		
Models	P	R	F1
CharBert+CharBert	0.518	0.457	0.473
BERT+BERT	**0.604**	0.445	0.481
ELECTRA+ELECTRA	0.546	**0.458**	**0.488**
BERT+ELECTRA	0.510	0.460	0.472
ELECTRA+CharBert	0.529	0.447	0.477
CharBert+ELECTRA	**0.531**	**0.485**	**0.492**

Table 4: Models Results in test dataset for BERT-related models in LCL setting. P and R stand for Precision and Recall separately.

Table 4 shows the results for different contextual encoders and weighting encoders we tried. The first term (Before "+" sign) is the model we used in the contextual encoder, the second one is what we used in the weighting encoder.

When we applied LCL on RoBERTa models (Such as BERTweet and CharBert-RoBERTa) as either contextual encoder or weighting encoder, an interesting result happened. The models only do classification on the top 3 or 4 high-frequency label but ignore others. We haven't figure it out why this would happen yet.

When two encoders use the same models, CharBert doesn't obtain a satisfying inprovement in terms of macro F1 score. BERT and ELECTRA obtain a roughly 1-2% increase in macro F1. Our explanation is that what did LCL do is actually adjusting the bias among labels. A good classification model like ELECTRA can solve this problems in some degrees. This results in the improvement for ELECTRA is not significant over BERT.

Using the same model for both contextual encoder and weighting encoder in LCL seems to bring some improvements in predicting fine-grained emotions. We are wondering what if we used different models for contextual encoder and weighting encoder. Under this thought, we tried several combinations of models. As a result, the CharBert+ELECTRA obtains the best F1 score.

When two encoders use different models, we acheive our best result when contextual encoder is CharBERT and weighting encoder is ELECTRA. In this setting, the model reaches a good balance between precision and recall. This is what we want to see in the results.

To better understand how the examples (sentences) are organized for classification compared with directly implementation and LCL, we apply t-SNE, a dimension reduction method that seeks to preserve distances between data points, using the scikit-learn package (Pedregosa et al. (2011)). We extract the hidden state vector for [CLS] token in the last hidden state for every example, then apply t-SNE for test dataset. The results can be explored in our scatter plot (See 3, 4 in Appendix A). We only show the examples that only have one output. Finally, we get 4590 samples in the test dataset. The color range from red to yellow to blue representing trends in negative, ambiguous, and positive emotions. The sentiment level is ordered by the author's preference. It's just a schematic plot. In a word, the LCL should tend to make the samples that have the same labels slightly closer than direct implementation, but we can not conclude a clear picture from these two figures. So, let's dive into fine-grained results to explore what did LCL do for improving the macro F1 score.

5.4.2 Fine-grained results

In fine-grained results, we mainly analyze the model performance for each emotion label, especially those low-frequency emotions.

Emotion	Precision	Recall	F1
admiration	0.65	0.63	0.64
amusement	0.76	0.82	0.78
anger	0.53	0.41	0.46
annoyance	0.35	0.35	0.35
approval	0.39	0.34	0.37
caring	0.39	0.33	0.32
confusion	0.39	0.45	0.42
curiosity	0.46	0.53	0.49
desire	0.54	0.34	0.41
disappointment	0.33	0.32	0.32
disapproval	0.37	0.37	0.37
disgust	0.43	0.49	0.46
embarrassment	0.43	0.41	0.42
excitement	0.40	0.39	0.40
fear	0.60	0.67	0.63
gratitude	0.93	0.90	0.91
grief	1.00	0.17	0.29
joy	0.54	0.65	0.60
love	0.75	0.83	0.79
nervousness	0.25	0.30	0.27
optimism	0.60	0.47	0.53
pride	0.67	0.50	0.57
realization	0.32	0.19	0.23
relief	0.38	0.27	0.32
remorse	0.59	0.86	0.70
sadness	0.56	0.52	0.54
surprise	0.61	0.55	0.58
neutral	0.66	0.54	0.60
micro-average	**0.57**	**0.53**	**0.55**
macro-average	**0.53**	**0.48**	**0.49**
std	**0.18**	**0.19**	**0.17**

Table 5: Results based on CharBert+ELECTRA for fine-grained emotions classification. **std** is the standard errors computed among all labels.

Table 5 shows the detailed performance of our best-implemented model, CharBert+ELECTRA, on the test dataset. This model obtains the best performance on emotions with overt lexical such as *gratitude*(.91), *amusement*(.78), *love*(.79), which has the similar pattern as BERT model. For the low frequency emotions, such as *greif*(.29), *relief*(.32), we got a better result than the baseline model. A surprising result is that our model achieve a highest precision (1.00) for the lowest frequency emotion grief. This is a good sign showing that the model starts to learn the feature of those low-frequency emotions. But there is still a room for us to improve.

	Comparison of F1 Scores		
Emotions	BERT	EL+EL	CB+EL
gratitude	0.91	**0.91**	0.91
amusement	0.75	**0.78**	0.76
love	**0.81**	0.75	0.79
grief	0.00	0.00	**0.29**
pride	0.40	**0.67**	0.56
relief	0.00	0.20	**0.32**
neutral	0.52	**0.62**	0.60
micro-average	0.54	**0.57**	0.55
macro-average	0.46	0.49	**0.49**

Table 6: Selected Results under different models. BERT is the baseline model we implemented. EL and CB stands for ELECTRA and CharBert separately. **Bold number** indicates the best performance among models. Particularly, EL+EL has largest F1 score for gratitude(0.9148) than CB+EL(0.9104) and BERT(0.9090). However, CB+EL has larger macro F1 score(0.4925) than EL+EL(0.4875).

Table 6 give us more detailed for comparison among the models. Our best performance model, CharBert+ELECTRA, outperformed other models in the lower frequency emotions such as grief and relief, which indicates LCL can help the model to better classify lower frequency emotions. However, CharBert+ELECTRA model has some performance loss in terms of high-frequency emotions like *amusement* etc. What's more, we only focus on an overall result sometimes, that is, treat the emotion label unequally. If the real data distribution is unequal for emotion labels, we would only consider an overall F1 score (micro F1). In this scenario, ELECTRA+ELECTRA model outperformed CharBert+ELECTRA model.

6 Conclusion

We investigate the performance of classic models on GoEmotions. The Char-CNN model performs not that well, which motivates us to find a better way to incorporate character-level information. When training the bi-LSTM model, the performance of the model is better when we train it with a fixed number of epochs. The resulting model of early stopping has worse performance. The results of bi-LSTM suggest that the plain Binary Cross-Entropy loss function may not be sufficient for the task.

We investigate the emotion classification task

based on BERT model. we try to improve the performance without changing model architecture significantly. The plain Binary Cross-Entropy loss has long been considered biased, especially in fine-grained classification task. Hence, we incorporate the Lable-aware Contrastive Loss (LCL) into GoEmotions classification tasks in order to reduce the influence of bias. We adjust a single-output LCL into a multi-output LCL. We also tried to incorporate character information into our model, for instance, CharBert. We tried different combination for contextual encoder and weighting encoder. As a result, we found that ELECTRA+ELECTRA model performs the best in the same model for contextual and weighting encoder setting, which has approximately 2.6%(0.488) increase in macro F1 score compared with our baseline BERT model(0.462). The CharBert+ELECTRA model performs the best(**0.492**) with approximately **3%** advance among all models in terms of macro F1 score. To dig behind the scenes, we investigate the detailed performances for all emotions. What did CharBert+ELECTRA did is improving the performance in low-frequency emotions but not losing too many performances in high-frequency emotions. We obtained a great improvement in low-frequency emotions such as *grief* (0.0 to 0.29) and *relief* (0.0 to 0.32) compared with baseline model BERT.

7 Contributions of Group Members

Haitao writes the paper reviews of Char-CNN, bi-LSTM and their variants. He implements Char-CNN and bi-LSTM on the given GoEmotions tasks. He observes that the Char-CNN performs significantly worse than other models and the bi-LSTM models show some interesting patterns during training.

Roy writes the paper reviews of GoEmotions datasets, BERT, BERT-related model in direct implementation and LCL setting. He implements various BERT and BERT-related models directly on the given GoEmotions tasks. He also implements the BERT and BERT-related model incorporating with LCL loss. He try different model for contextual encoder and weighting encoder. He do the visualization via t-SNE for representative vector of [CLS] token in the last hidden state. He creates figures for illustrating LCL structure and tables for reporting results. He observes that the LCL setting works well in find-grained emotions classification task through improving F1 score for those low frequency emotions.

References

Thérese Bergsma, Judith van Stegeren, and Mariët Theune. 2020. Creating a sentiment lexicon with game-specific words for analyzing npc dialogue in the elder scrolls v: Skyrim. In *Workshop on Games and Natural Language Processing*, pages 1–9.

Erik Boiy and Marie-Francine Moens. 2009. A machine learning approach to sentiment analysis in multilingual web texts. *Information retrieval*, 12(5):526–558.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. *CoRR*, abs/2003.10555.

Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. 2020. GoEmotions: A Dataset of Fine-Grained Emotions. In *58th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Cicero Dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th international conference on computational linguistics: technical papers*, pages 69–78.

Michael Gamon, Sumit Basu, Dmitriy Belenko, Danyel Fisher, and Matthew Hurst. 2008. Blews: Using blogs to provide context for news articles. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 2, pages 60–67.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. *CoRR*, abs/2004.11362.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Wentao Ma, Yiming Cui, Chenglei Si, Ting Liu, Shijin Wang, and Guoping Hu. 2020. Charbert: Character-aware pre-trained language model. *arXiv preprint arXiv:2011.01513*.

Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. BERTweet: A pre-trained language model for English Tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Kurt Shuster, Eric M Smith, et al. 2020. Recipes for building an open-domain chatbot. *arXiv preprint arXiv:2004.13637*.

Sara Rosenthal, Alan Ritter, Preslav Nakov, and Veselin Stoyanov. 2014. SemEval-2014 task 9: Sentiment analysis in Twitter. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 73–80, Dublin, Ireland. Association for Computational Linguistics.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.

Carlo Strapparava and Rada Mihalcea. 2007. SemEval-2007 task 14: Affective text. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 70–74, Prague, Czech Republic. 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. *CoRR*, abs/2109.05427.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

A t-SNE results

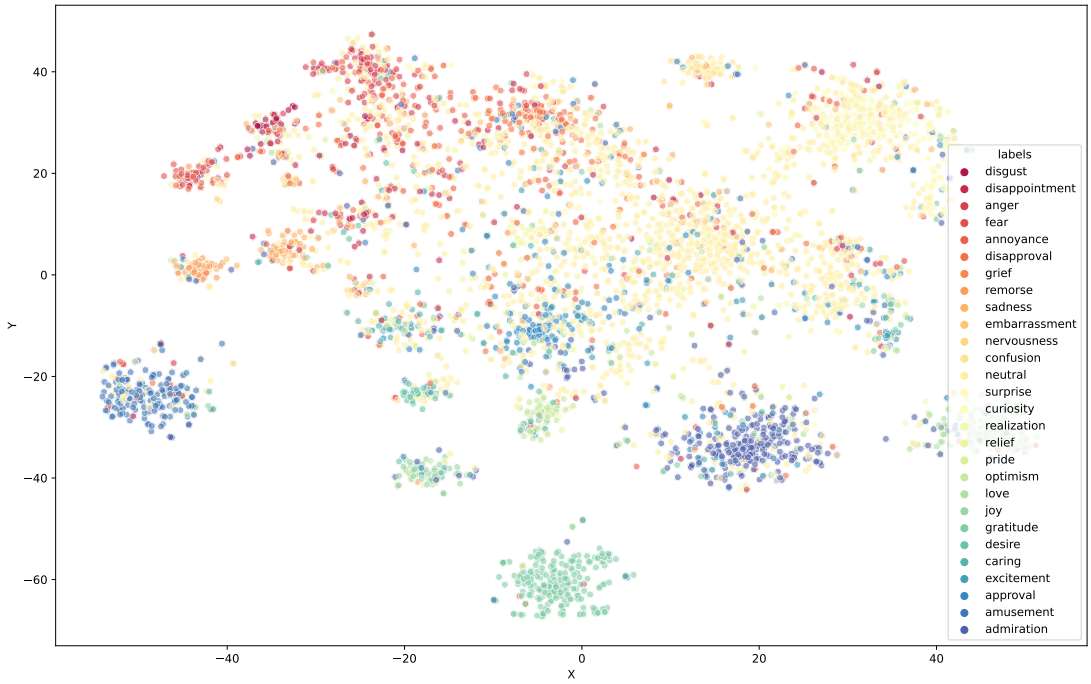

Figure 3: t-SNE result for BERT directly implementation

Figure 4: t-SNE result for BERT with LCL