# Multi-Labeled Groups Detection and Classification of Emotions in Texts Emotional Damage Team

**Tao Zhang, Yuxuan Wang, Bowei Wu, Jiahui He, Xichen Li**
{tao.zhang1,yuxuan.wang1,bowei.wu,jiahui.he,xichenli}@ufl.edu

## Abstract

As one of the main research problems in natural language processing, text sentiment analysis uses natural language processing and text mining techniques to analyze, process, generalize, and reason about subjective texts with emotional overtones, and finally arrive at emotional categories. The goal of this project is to fine-tune the training on the processed micro sentiment multi-classification dataset GoEmotions by pre-training model Erinie 3.0 to build a refined micro sentiment multi-classification model to refine the granularity of sentiment analysis results while optimizing the micro sentiment classification effect.

## 1 Introduction

### 1.1 Background

Due to the Internet's recent rapid expansion, people may now express themselves more readily online. It is crucial to learn how to instantly recognize ideas, attitudes, and feelings in writing since the prevalence of subjective expression is growing. even if the majority of research is done from a public or business perspective(Li and Ren, 2009).In truth, there is a growing amount of understanding about how individuals feel, so it is important to pay attention to a person's emotional state. The processing of multiple sentiment information has become an increasingly challenging research field(Wang and Cao, 2020).

For instance, in the area of social media, organizations can utilize multiple sentiment analyses to learn more about the perception of their brand by examining social media posts and customer reviews. Businesses can better their goods and services by examining various data sources to determine the customers' favorable and negative attitudes(Micu et al., 2017).

One further instance is in the area of customer service, where it is possible to employ multiple sentiment analyses to examine client feedback and complaints. Businesses may be able to better serve their customers and enhance their goods and services by using this information to spot trends and problems(Luo et al., 2021).

Overall, multiple sentiment analysis with NLP offers a powerful tool for comprehending the sentiments and opinions represented in text data, empowering businesses and organizations to make wise decisions. The significance of this method cannot be emphasized in the modern data-driven environment, where success depends on having the capacity to draw insightful conclusions from massive amounts of data(Mejova, 2009).

### 1.2 GoEmotions

The GoEmotions(Demszky et al., 2020) micro-sentiment multi-categorization dataset differs from the typical sentiment 2-7 classification in that it increases sentiment granularity to 28 types for the first time. In contrast to simple single classification tasks, a sentence in the GoEmotions dataset may correlate to two or even three feelings, indicating a text multi-label classification job. This project will show how to fine-tune the ERNIE 3.0 pre-training model, which is built on PaddleNLP, in order to perform GoEmotions micro-emotion text multi-label classification prediction.

### 1.3 Erinie 3.0

Ernie 3.0(Sun et al., 2021) is pre-trained on a large-scale Chinese and English corpus and fine-tuned on several downstream tasks to improve its performance. It incorporates several knowledge-enhancement techniques, such as knowledge distillation and knowledge masking, to improve its ability to reason and generalize across different tasks.

### 1.4 PaddleNLP

PaddleNLP is an easy-to-use and powerful NLP library with an Awesome pre-trained model zoo, sup-

porting a wide range of NLP tasks from research to industrial applications. It has some features such as Out-of-Box NLP Toolset, Industrial End-to-end System, High-Performance Distributed Training and Inference, etc.

### 1.5 Problem Statement

In sentiment analysis, granularity and accuracy are important factors that affect the effectiveness of sentiment analysis. Most existing sentiment analysis systems currently use positive, neutral, and negative sentiment classification, which is harder to express the complexity of human emotions and difficult to tap the potential emotions in text. 2020 ACL conference Google researchers released GoEmotions, the largest and most fine-grained micro-emotion manually annotated dataset to date, containing 58,000 manually annotated Reddit comments(Alm et al., 2005), with the emotion category raised to 28 for the first time, providing an opportunity to better tap into users' underlying sentiment.

This project will fine-tune the training on the processed micro-sentiment multi-categorization dataset GoEmotions with the pre-training model Erinie 3.0 to build a refined micro-sentiment multi-categorization model, refining the granularity of sentiment analysis results while optimizing the micro-sentiment classification effect.

## 2 Dataset Preprocessing

### 2.1 Data Preprocessing

For the 28 micro-sentiment multi-label classification scenarios(Demszky et al., 2020), where a sentence may correspond to multiple sentiment category labels, the sentiment labels of the dataset need to be transformed using One-Hot coding first, with 0 indicating absence and 1 indicating presence for each sentiment. The 28 types of micro-emotional mapping relationships are shown below in Table 1.

Table 1: 28 types of micro-emotional mapping relationships

| | | | |
|---|---|---|---|
| 1 | admiration | 2 | amusement |
| 3 | anger | 4 | annoyance |
| 5 | approval | 6 | caring |
| 7 | confusion | 8 | curiosity |
| 9 | desire | 10 | disappointment |
| 11 | disapproval | 12 | disgust |
| 13 | embarrassment | 14 | excitement |
| 15 | fear | 16 | gratitude |
| 17 | grief | 18 | joy |
| 19 | love | 20 | nervousness |
| 21 | neutral | 22 | optimism |
| 23 | pride | 24 | realization |
| 25 | relief | 26 | remorse |
| 27 | sadness | 28 | surprise |

### 2.2 Loading pre-trained models ERNIE 3.0

ERNIE 3.0 introduces a large-scale knowledge graph in the tens of billions pre-training model for the first time and proposes a parallel pre-training method of massive unsupervised text and large-scale knowledge graph (Universal Knowledge-Text Prediction), by inputting the 50 million knowledge graph triples obtained by the knowledge graph mining algorithm and 4TB large-scale corpus into the pre-training model at the same time. This promotes information sharing between structured knowledge and unstructured text, and significantly improves the model's ability to remember and reason about knowledge.

The ERNIE 3.0 framework(Sun et al., 2021) shown in figure 1 is divided into two layers. The first layer is the generic semantic representation network, which learns the basic and generic knowledge in the data. The second layer is the task semantic representation network, which learns task-related knowledge based on the generic semantic representation. During the learning process, the task semantic representation network learns only the pre-trained tasks of the corresponding category, while the generic semantic representation network learns all the pre-trained tasks.

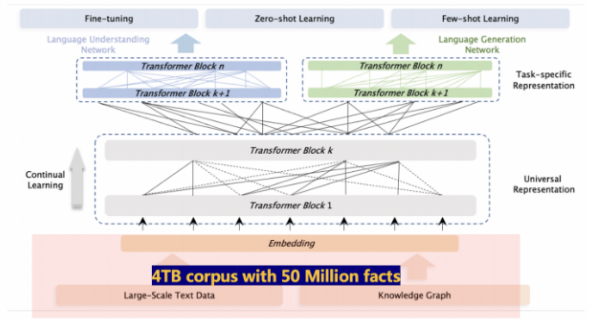


Figure 1: Ernie 3.0 framework structure

### 2.3 Data processing into model-acceptable formats

The dataset is usually raw data that requires some data processing and sampling group batch: the dataset is processed from raw text into input for the model using a word splitter through Dataset's map function.

## 3 Training Model

The field of natural language processing has seen rapid advancement because of the advent of deep learning, which has led to the emergence of numerous high-quality pre-training models that are similar to Transformer.(Rath et al., 2022) These models have continuously updated SOTA (State of the Art) for various NLP applications. Users can access popular pre-trained models from PaddleNLP along with the weights that correlate to them. Developers can quickly and easily apply various Transformer class pre-training models and their downstream tasks using tools like BERT, ERNIE, ALBERT, RoBERTa, XLNet, etc., which use a unified API for loading, training, and invocation. The corresponding pre-training model weights also download quickly and steadily(Gillioz et al., 2020).

A straightforward and user-friendly interface is provided by the Auto module of PaddleNLP (which includes AutoModel, AutoTokenizer, and numerous downstream task classes) to call pre-trained models of various network configurations without specifying model classes. The $from\_pretrained()$ method of PaddleNLP makes it simple to load pre-trained models, and the Transformer pre-trained model overview includes more than 40 of the most popular pre-trained models with more than 500 model weights. AutoModelForSequenceClassification can be used for multi-label classification by pre-training the model to obtain a representation of the input text, after which the text representation is classified.

### 3.1 Optimizer Selection

For the training process of this model, we chose to use the AdamW optimizer, the cross-on-loss function, and a custom MultiLabelReport evaluation metric. Here is the formula for the AdamW optimizer.

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{1}$$

where $\theta_t$ is the parameter at time $t$, $\alpha$ is the learning rate, $\hat{m}_t$ is the biased first moment estimate, $\hat{v}_t$ is the biased second raw moment estimate, and $\epsilon$ is a small constant to prevent division by zero.

The $\hat{m}_t$ and $\hat{v}_t$ are updated as follows:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \tag{2}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} + \frac{\lambda}{\sqrt{1 - \beta_2^t}}, \tag{3}$$

where $m_t$ and $v_t$ are the first and second raw moment estimates, $\beta_1$ and $\beta_2$ are the decay rates for the two moments, and $\lambda$ is the weight decay parameter. Compared to the Adam optimizer, here is the formula for the Adam optimizer,

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{4}$$

where $\theta_t$ is the parameter at time $t$, $\alpha$ is the learning rate, $\hat{m}_t$ is the biased first moment estimate, $\hat{v}_t$ is the biased second raw moment estimate, and $\epsilon$ is a small constant to prevent division by zero.

The $\hat{m}_t$ and $\hat{v}_t$ are updated as follows:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \tag{5}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}, \tag{6}$$

where $m_t$ and $v_t$ are the first and second raw moment estimates, $\beta_1$ and $\beta_2$ are the decay rates for the two moments, respectively(Llugsi et al., 2021). Adam adjusts the learning rate automatically, which significantly speeds up training and only occasionally necessitates changing the learning rate. Due to the accumulation of dividing by the squared gradient throughout the Adam computing phase, the subtraction term will be tiny. Common sense dictates that heavier people should be punished more severely, yet Adam does not follow this logic. The penalty is plainly greater for

heavier weights because weight decay is updated using the same factors for all weights. AdamW, which is Adam + weight decade, has the same effect as Adam + L2 regularization but is more effective because AdamW directly adds the regular term to the backpropagation formula, whereas L2 regularization first calculates the gradient, then backpropagates(Sun and Chen, 2022). Here is how L2 regularization works,

$$L_{reg} = \frac{\lambda}{2} \sum_{i=1}^{n} ||w_i||_2^2 \tag{7}$$

where $L_{reg}$ is the regularization loss, $\lambda$ is the regularization strength hyperparameter, $n$ is the number of weights in the model, and $w_i$ is the weight associated with the $i$-th parameter in the model. The $||w_i||_2^2$ term represents the squared L2 norm (also known as Euclidean norm) of the weight $w_i$. This eliminates the need to manually add the regular term to the loss.

### 3.2 Loss Function Selection

When neural networks are employed to solve classification issues, the cross-entropy loss function is frequently used, and cross-entropy itself is frequently used as a loss function(Rezaei-Dastjerdehei et al., 2020). We employ the BCEWithLogitsLoss approach in this training. By computing the binary cross entropy with logits loss between the input logit and the label, this method provides a callable class of BCEWithLogitsLoss. The sigmoid operation and the api_nn_loss_BCELoss operation are combined in this approach. Additionally, we may think of this approach as combining some reduced operations with sigmoid_cross_entrop_with_logits. This OP can determine the probability error for each category-independent classification task per element. One way to conceptualize it is as data point label prediction where the labels are not mutually exclusive(Li et al., 2020). An article of news, for instance, might be about politics, technology, sports, or none of these things at once. It is true that for the data set we are using, a sentence may contain the emotions "happy", "excited" and "unbelievable" at the same time.

The learning rate and the bias value, when used in conjunction with gradient descent parameter updates, determine how quickly a model learns. We concentrate on the bias value since, among them, the learning rate is the hyperparameter we need to set. We discover that x_i and $[\sigma(s) - y]$ have an impact on the magnitude of the bias value(Konar et al., 2020). We pay particular attention to the latter, whose magnitude value indicates the degree of error in our model; the larger the value, the worse the model; however, the larger the value also makes the bias value larger, and thus the model learns faster. As a result, when utilizing the logistic function to calculate probability and the cross-entropy as the loss function, learning progress is made more quickly when the model is weak and more slowly when it is strong.

### 3.3 Evaluation Index Selection

Prediction outcomes depend on precision. The precision of the model is reflected by the probability that the prediction in the sample with positive prediction is true, which is comparable to the proportion of correct answers a candidate submitted on the exam paper. Additionally, recall shows the percentage of the genuine positive samples that were deemed to be positive, which is comparable to how many questions a candidate correctly answered on the exam. In general, Precision and Recall are mutually exclusive metrics. When Precision is high, Recall tends to be low, and vice versa when Precision is low. The two measures can be taken into account simultaneously because the main goal of F1 is to maximize Precision and Recall while minimizing the discrepancy between them. Macro-average counts the number of TP, FP, FN, and TN items in each category, determines the corresponding Precision and Recall values to produce the corresponding F1 values, and then averages the results to provide Macro-F1. In order to evaluate the performance of the model in a more thorough manner, we, therefore, combine different measures when establishing assessment metrics.

## 4 Metrics Selection

Metric selection is an essential step for evaluating NLP tasks(Yang and Liu, 1999). Different metrics focus on different aspects of model performance, and the choice of a metric depends on the specific goals of the task and other factors like the data set.

An inappropriate selection of metrics can lead to misleading results and misinterpretation of model performance. A model with a high F1 score may still get poor performance in terms of recall or precision, which depends on the distribution of positive and negative examples in the data set.

To make sure that our NLP model is evaluated

accurately and effectively, it is essential to carefully analyze the data set and our requirements for the task.

### 4.1 Common metrics in NLP

Nowadays, there are many metrics available to measure the performance of NLP models. These metrics include accuracy, precision, recall, F1-score, area under the receiver operating characteristic curve (AUC-ROC), the area under the precision-recall curve (AUC-PR), and mean squared error (MSE). By selecting the appropriate metrics for a specific NLP task, the performance of the model can be evaluated effectively, and making informed decisions regarding its optimization and implementation.

### 4.2 Explanation of selected metrics

Our task is a multi-label sentiment analysis problem, so it does not make sense to use evaluation metrics applied to information retrieval or text summarization.

Although accuracy is the most commonly used evaluation metric, after studying the GoEmotions dataset, we found that there are large unbalanced samples in GoEmotions. We believe that accuracy does not correctly represent the performance of our model. Since not all 28 labels have balanced positive and negative samples, in other words, there are some labels with far more positive samples than negative samples or vice versa.

The 28 micro-emotions were modeled as 0 and 1, where 0 means that the sentiment is not present in the sentence, and 1 means that it is present. For a small number of sentiments, the number of negative samples far exceeds the number of positive samples, assuming that the model all predict negative samples 0 at this time can still achieve a high accuracy rate, the model will lose the ability to identify positive samples, and the high accuracy rate does not reflect the predictive ability of the model. Therefore, it is more suitable to use Precision, Recall, and F1-Score for comprehensive evaluation of multi-categorization. Based on this analysis, a model that achieves high accuracy may still perform poorly on a particular task because accuracy does not take into account the relative importance of different types of errors, so we do not use accuracy in our metric.

**Precision**: The proportion of related instances out of the total predicted instances. It measures how well the model avoids false positives.

**Recall**: The proportion of relevant instances to the total number of actual instances. It measures how well the model avoids false negatives.

**F1-score**: The harmonic mean of precision and recall, which provides a single score that balances both metrics.

There are two types of F1 scores that are commonly used in multi-label classification tasks: micro-F1 and macro-F1(Sokolova and Lapalme, 2009).

**Micro-F1** gives equal weight to each instance, regardless of which class it belongs to. This is particularly useful in imbalanced datasets, where some classes may have very few instances compared to others. In such cases, treating each class separately could lead to misleading evaluation metrics. By aggregating the counts across all classes, Micro-F1 provides a more accurate measure of the overall performance of the model.

$$MicroF1 = \frac{2 \times (m - prec \times m - recall)}{m - prec + m - recall} \tag{8}$$

where:

$$micro - precision = \frac{TP}{TP + FP} \tag{9}$$

$$micro - recall = \frac{TP}{TP + FN} \tag{10}$$

**Macro-F1** gives equal weight to each class, regardless of the number of instances in each class. This is particularly useful in datasets where each class is equally important, or when the distribution of instances across classes is roughly balanced. By treating each class separately, Macro-F1 provides a more nuanced evaluation of the model's performance for each class.

$$Macro - F1 = \frac{1}{n} \sum_{i=1}^{n} F1_i \tag{11}$$

where:

$$F1_i = 2 \times \frac{precision_i \times recall_i}{precision_i + recall_i} \tag{12}$$

$$precision_i = \frac{TP_i}{TP_i + FP_i} \tag{13}$$

$$recall_i = \frac{TP_i}{TP_i + FN_i} \tag{14}$$

**AUC**: Usually used in classification tasks, it measures the model's ability to distinguish positive and negative examples at different thresholds.

### 4.3 Analysis of metrics

Precision is for prediction results. The probability that the prediction is correct in the sample with positive prediction is similar to how many answers a candidate wrote in the exam paper are correct, which reflects the precision of the model. And Recall indicates the proportion of the actual positive samples judged to be positive, similar to how many questions a candidate answered in the exam paper. Precision and Recall are contradictory measures, generally speaking, when Precision is high, Recall tends to be low; while when Precision is low, Recall tends to be high.

The core idea of F1 is to improve Precision and Recall as much as possible while keeping the difference between them as small as possible, and the two metrics can be considered together. Macro-average counts TP, FP, FN, and TN of each category, calculates the respective Precision and Recall to get the respective F1 values, and then takes the average to get Macro-F1.

## 5 Training Experiment and Evaluation

### 5.1 Loss Function Experiments

In each training step (iteration) of the network, the results computed in the forward propagation step are compared with the ground truth (actual) values to compute the error or loss fraction. This error is then back-propagated in the back-propagation step to adjust the initial values of the weights and biases so that the error is minimized in the next training steps. We continue training until we obtain the minimum error.

To calculate the error between the predicted and ground truth (actual) values in each training step, a loss function is required. When we take the training experiments of the loss function, we try to call different loss functions based on the paddle.nn package. As we said, most of the neural network models use cross-entropy as the loss function. We first take an experiment of our model training with the CrossEntropyLoss function, but we found the parameters that paddle.nn.The CrossEntropyLoss function used is not adapted to our set-up labels. Besides trying with paddle.nn.CrossEntropyLoss function, we also tried paddle.nn.CosineEmbeddingLoss function, paddle.nn.CTCLoss, and paddle.nn.BCELoss function. And these three loss functions could not adapt to our set-up labels as well. We finally decided to take experiments with two adapted loss functions, which are paddle.nn.BCEWithLogitsLoss and paddle.nn.MSELoss function, because the BCELosswithLogits function could compute the binary cross entropy with logits loss between the input logit and the set-up label, and the MSEloss could compute the mean square error of given inputs and labels. Then we took two training experiments by setting the learning rate between 0.06 to 0.08 and using these two loss functions to compare the training accuracy and to find the more appropriate loss function we could use in our multi-emotion labeled-groups classification. The preliminary results of the training accuracy of these two loss functions are shown below in Table 2.

Table 2: Results of Training Accuracy with Loss Function Experiments

| Epoch=3 | BCEWithLogitsLoss | MSELoss |
|---|---|---|
| Eval Loss | 1.80687 | 0.35317 |
| AUC | 0.68974 | 0.67353 |
| F-1 Score | 0.30401 | 0.30401 |
| Precision | 0.32928 | 0.32928 |
| Recall | 0.28235 | 0.28235 |

With evaluation results about the loss function, the eval loss of BCEWithLogitsLoss is higher than the loss of MSELoss, but the eval loss of BCEWithLogitsLoss is even greater than 1, which is unreasonable. The reason could be due to some parameters or some layers in the training model. And we found after we take experiments with another learning rate, the eval loss of BCEWithLogitsLoss is in a rational range. Therefore, with the same F-1 score, Precision, and Recall score, our comparison is mainly focused on the AUC score of two training with these two loss functions. Then we determine to select BCEWithLogitsLoss as our model's loss function since training with BCEWithLogitsLoss has a higher AUC score.

### 5.2 Optimizer Selection and Configuration Running

During the training process, our goal is to minimize the loss function as much as possible. In order to minimize the loss function, we need an optimizer (optimization algorithm).

After determining evaluation metrics, we define a class named, MultiLabelReport, so that we could calculate all these different metrics together with every iteration's figures in the model's training.

These metrics include AUC_score, F1-score, precision, and recall.

Then we need to select the optimizer and we take a comparison of model training and validation with three different optimizers, which are Adam Optimizer, Adamax Optimizer, and AdamW Optimizer.

The preliminary results of the training accuracy of these three optimizers are shown below in Table 3.

Table 3: Results of Training Accuracy with Optimizer Experiments

| Epoch=3 | Adam | Adamax | AdamW |
|---|---|---|---|
| Eval Loss | 0.14870 | 0.13739 | 0.08618 |
| AUC | 0.76312 | 0.80681 | 0.95079 |
| F-1 Score | 0.30410 | 0.39162 | 0.60440 |
| Precision | 0.32904 | 0.44919 | 0.58179 |
| Recall | 0.28267 | 0.34713 | 0.62885 |

From the validation results, we can see that the AdamW optimizer would be the best option among these three optimizers. Then we select to use AdamW optimizer as our optimizer.

### 5.3 Learning Rate Experiments

The learning rate is the most important hyperparameter in a neural network and it is used in the optimization algorithm, such as Adam optimizer. At the beginning of the training process, the parameters of the network (weights and biases) are initialized with random values, but these are not the best values that give the minimum loss since we want to lower the loss function with an optimizer. That is the reason why we have to continue the training process further. Since we have already found BCEWithLogitsLoss function would fit in our model training with lower loss, we would keep on fine-tuning the learning rate to find which learning rate would have a lower loss.

For our experiments about learning rate, we conduct several times of model training by setting up the learning rate as 0.08, 0.005, and 0.00004 separately and comparing the evaluation metrics of different training with these learning rates. Meanwhile, we keep on setting up the weight decay as 0.01.

The preliminary results of the training accuracy of these three different learning rates are shown below in Table 4.

Table 4: Results of Training Accuracy with Learning Rate Experiments

| Learning Rate | 0.08 | 5e-3 | 4e-5 |
|---|---|---|---|
| Eval Loss | 1.80687 | 0.18077 | 0.08618 |
| AUC | 0.68974 | 0.73482 | 0.95079 |
| F-1 Score | 0.30401 | 0.30401 | 0.60440 |
| Precision | 0.32928 | 0.32928 | 0.58179 |
| Recall | 0.28235 | 0.28235 | 0.62885 |

From the results, we know when we set the learning rate as 0.00004, the model training gets a higher accuracy of AUC, F-1 Score, Precision score, Recall score and a lower loss. We would try to compare other learning rates for our future scope to see if other smaller learning rate could get a higher accuracy.

### 5.4 Batch Size Experiments

The batch size is a hyperparameter that defines the number of samples to be processed before updating the internal model parameters.

A batch can be thought of as a for-loop that iterates over one or more samples and makes predictions. at the end of the batch, the predictions are compared to the expected output variables, and an error is calculated. From this error, the update algorithm is used to improve the model, for example, by moving down the error gradient.

A training dataset can be divided into one or more batches. We split the dataset into the training dataset and the test dataset. To compare the effect of different batch sizes on our model's training accuracy, we conduct two groups of batch sizes on the training dataset and the test dataset. In the first group of training hyperparameters, we set the batch size of the training dataset as 32 and set the batch size of the test dataset as 16. In the second group of training hyperparameters, we set the batch size of the training dataset as 64 and set the batch size of the test dataset as 32.

The preliminary results of the training accuracy of these two groups of batch sizes on the train dataset and test dataset are shown below in Table 5.

Table 5: Results of Training Accuracy with Batch Sizes Experiments

| Epoch = 3 Batch Sizes | Train dataset=32 Test dataset=16 | Train dataset=64 Test dataset=32 |
|---|---|---|
| Eval Loss | 0.08618 | 0.08904 |
| AUC | 0.95079 | 0.94519 |
| F-1 Score | 0.60440 | 0.59322 |
| Precision | 0.58179 | 0.57435 |
| Recall | 0.62885 | 0.61337 |

From the results, we can find that doubling the batch sizes on the training dataset and the test dataset does not improve the training accuracy.

### 5.5 Epoch Times Experiments

The number of epochs is a hyperparameter that defines the number of times the learning algorithm works on the entire training dataset. An epoch means that each sample in the training dataset has the opportunity to update the internal model parameters. An epoch is composed of one or more batches. As a for-loop on the number of epochs, with each loop being performed on the training dataset, there is also an embedded for-loop within this for-loop that iterates over each batch of samples, where the batch has a specified number of "batch sizes".

Since the model takes a long period of time on training, we conduct experiments on Epoch Times 1, 3, and 6 separately. The preliminary results of the training accuracy of these three sets of epoch times are shown below in Table 6.

Table 6: Results of Training Accuracy with Epoch Times Experiments

| Epoch Times | 1 | 3 | 6 |
|---|---|---|---|
| Eval Loss | 0.09662 | 0.08618 | 0.08623 |
| AUC | 0.92884 | 0.95079 | 0.95069 |
| F-1 Score | 0.56616 | 0.60440 | 0.60178 |
| Precision | 0.55569 | 0.58179 | 0.58872 |
| Recall | 0.57703 | 0.62885 | 0.61542 |

From the results of training accuracy with different epoch times, we can see the model training with either Epochs as 3 or Epochs as 6 get similar performance on evaluation metrics. The f-1 score and Recall score of model training with epochs as 3 are a little higher than the training with epochs as 6. However, the Precision score of model training with epoch 3 is a little lower than it with epoch 6. Therefore, we need to determine the epochs from the prediction results of our model.

### 5.6 Prediction Results Analysis

Since model training with epoch 3 and epoch 6 has similar training performances, we would check the results of two models' predictions that are trained with epochs times 3 and 6.

We write 28 types of micro-emotional texts as our predicted data first. Then we use the trained ERNIE 3.0 Model with both epochs as 3 and 6 to predict what type of emotion group these multiple micro-emotional texts should be classified to. And we check how accurately these texts are classified into corresponding emotion-labeled groups by our trained models.

In figure 2, the prediction results of the trained model with Epochs as 3 show that the annoyance type of emotional text is misclassified, some micro-emotional groups of texts are classified as neutral groups and some texts cannot be recognized.

```
Text: You do a great job!        Labels: admiration
Text: Lets have fun!     Labels: joy
Text: You shut your mouth        Labels: anger
Text: You are so annoyed         Labels: anger
Text: You are allowed to do this         Labels: neutral
Text: Are you feeling well?      Labels: curiosity
Text: This problem is so hard and I cannot solve this problem    Labels:
Text: Why would I do that?       Labels: curiosity
Text: I want this gift so much   Labels: desire
Text: I am very disappointed by everything you have done to me   Labels: disappointment
Text: You are not admitted to the college.       Labels:
Text: Thats absolutely disgusting.       Labels: disgust
Text: Thats so embarrassing.      Labels: embarrassment
Text: I am so excited     Labels: excitement
Text: I am so scared of skydiving        Labels: fear
Text: Thank you.         Labels: gratitude
Text: My grandpa passed away      Labels: sadness
Text: Happy Birthday     Labels:
Text: I love you so much         Labels: love
Text: I am so nervous.   Labels: nervousness
Text: It is just so so.          Labels: neutral
Text: Successful people only focus on giving their best effort.          Labels:
Text: I am so proud of you.      Labels: admiration,pride
Text: Thank you for letting me realizing this rule.       Labels: gratitude
Text: You are doing better than you think you are        Labels:
Text: I am guilty.       Labels: remorse
Text: I am so sad.        Labels: sadness
Text: I am so surprised that you made it.        Labels: surprise
```

Figure 2: Prediction Results of Model with Epochs as 3

However, in figure 3, the prediction results of the trained model with Epochs as 6 show that there are fewer misclassified texts and fewer micro-emotional texts are classified into neutral groups. Although there are a few texts that cannot be recognized, this means we need to keep on improving the training performance and the accuracy of classification.

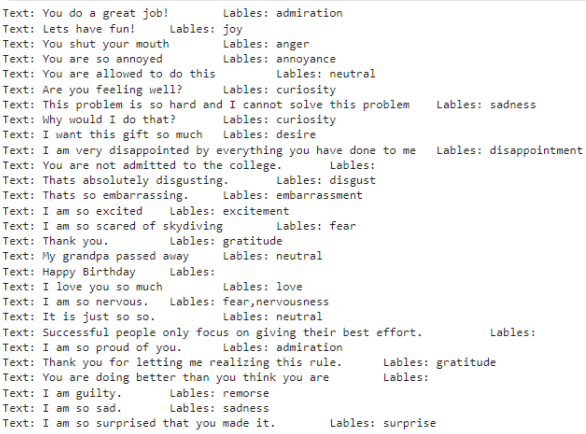


Figure 3: Prediction Results of Model with Epochs as 6

# 6 Conclusion and Future Work

## 6.1 Conclusion

In our project, we aim to solve the problem that traditional existing sentiment analysis models can only recognize neutral, positive, and negative emotions in texts. In this way, we find a pre-trained model, ERNIE 3.0 from the Paddle package, and try to use this model to recognize at most 28 micro-emotions categorized by the GoEmotions dataset and classify random texts into 2 to 3 corresponding emotion groups. Moreover, we keep on conducting experiments of fine-tuning parameters in ERNIE 3.0 Model training to refine the model which can recognize the most emotion-labeled groups from texts with a higher accuracy.

To summarize the outline of our project, we first preprocess the dataset and convert it to applicable input for ERNIE 3.0 Model. Then we load the ERNIE 3.0 Model and determine the validation metrics to evaluate our training's performance. The next part is ERNIE 3.0 Model Training. Our experiments are using different optimizers and loss functions, fine-tuning the learning rate, the batch sizes on the train and test datasets, and epoch times to compare the training accuracy with these various elements.

In conclusion, based on our prediction results, after fine-tuning the parameters in our model training, we determined to use the BCELosswithLogits as the loss function and AdamW as the optimizer. By results of experiments with learning rate and batch sizes, we set the learning rate as 0.00004 and set the training dataset's batch size as 32 and the test dataset's batch size as 16. After comparing the prediction results, we select 6 as the Epoch Times of the ERNIE 3.0 Model. Based on our metrics with these parameters, the model has an average F-1 score of 0.60178 and an average Recall score of 0.6154. According to our prediction results with these parameters and functions, our model could detect 20-22 micro-emotions and recognize two or three different emotions combined in a text at present.

## 6.2 Future Work

We will continue our experiments to explore ways that can improve performance. We will try to find other models that can recognize emotions in texts and can classify them into different emotion-labeled groups with higher accuracy. We could improve the linguistic learning of ERNIE 3.0 to recognize and classify emotions in different languages by considering these languages' grammar and semantics meanings.

# References

Cecilia Ovesdotter Alm, Dan Roth, and Richard Sproat. 2005. Emotions from text: Machine learning for text-based emotion prediction. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 579–586, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. 2020. Goemotions: A dataset of fine-grained emotions.

Anthony Gillioz, Jacky Casas, Elena Mugellini, and Omar Abou Khaled. 2020. Overview of the transformer-based models for nlp tasks. In *2020 15th Conference on Computer Science and Information Systems (FedCSIS)*, pages 179–183.

Jinia Konar, Prerit Khandelwal, and Rishabh Tripathi. 2020. Comparison of various learning rate scheduling techniques on convolutional neural network. In *2020 IEEE International Students' Conference on Electrical,Electronics and Computer Science (SCEECS)*, pages 1–5.

Huana Li and Fuji Ren. 2009. The study on text emotional orientation based on a three-dimensional emotion space model. In *2009 International Conference on Natural Language Processing and Knowledge Engineering*, pages 1–6.

Li Li, Miloš Doroslovački, and Murray H. Loew. 2020. Approximating the gradient of cross-entropy loss function. *IEEE Access*, 8:111626–111635.

Ricardo Llugsi, Samira El Yacoubi, Allyx Fontaine, and Pablo Lupera. 2021. Comparison between adam, adamax and adam w optimizers to implement a

weather forecast based on neural networks for the andean city of quito. In *2021 IEEE Fifth Ecuador Technical Chapters Meeting (ETCM)*, pages 1–6.

Jian Ming Luo, Huy Quan Vu, Gang Li, and Rob Law. 2021. Understanding service attributes of robot hotels: A sentiment analysis of customer online reviews. *International Journal of Hospitality Management*, 98:103032.

Yelena Mejova. 2009. Sentiment analysis: An overview. *University of Iowa, Computer Science Department*.

Adrian Micu, Angela Eliza Micu, Marius Geru, and Radu Constantin Lixandroiu. 2017. Analyzing user sentiment in social media: Implications for online marketing strategy. *Psychology & Marketing*, 34(12):1094–1100.

Asutosh Rath, B. Hridaya, Duggaraju Vimala, and Jossy George. 2022. Multilingual sentiment analysis of youtube live stream using machine translation and transformer in nlp. In *2022 International Conference on Trends in Quantum Computing and Emerging Business Technologies (TQCEBT)*, pages 1–5.

Mohammad Reza Rezaei-Dastjerdehei, Amirmohammad Mijani, and Emad Fatemizadeh. 2020. Addressing imbalance in multi-label classification using weighted cross entropy loss function. In *2020 27th National and 5th International Iranian Conference on Biomedical Engineering (ICBME)*, pages 333–338.

Marina Sokolova and Guy Lapalme. 2009. A systematic analysis of performance measures for classification tasks. *Information Processing Management*, 45(4):427–437.

Yajing Sun and Aixiang Chen. 2022. An adaptive gradient descent optimization algorithm based on stratified sampling. In *2022 3rd International Conference on Computer Vision, Image and Deep Learning International Conference on Computer Engineering and Applications (CVIDL ICCEA)*, pages 1225–1231.

Yu Sun, Shuohuan Wang, Shikun Feng, Siyu Ding, Chao Pang, Junyuan Shang, Jiaxiang Liu, Xuyi Chen, Yanbin Zhao, Yuxiang Lu, Weixin Liu, Zhihua Wu, Weibao Gong, Jianzhong Liang, Zhizhou Shang, Peng Sun, Wei Liu, Xuan Ouyang, Dianhai Yu, Hao Tian, Hua Wu, and Haifeng Wang. 2021. Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation.

Zhou Wang and Jing Cao. 2020. Multi-task learning network for document-level and multi-aspect sentiment classification. In *2020 IEEE Fifth International Conference on Data Science in Cyberspace (DSC)*, pages 171–177.

Yiming Yang and Xin Liu. 1999. A re-examination of text categorization methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–49.