Sentiment Recognition in Conversations Project Report

Xiong Yang 1004876
Gao Fancheng 1004879

June 14, 2023

1 Introduction

Emotion recognition in conversations (ERC) refers to the ability to identify and understand the emotions expressed by one or more people during a conversation. This can be done by analyzing the tone of voice, facial expressions, body language, and the words and phrases used by the speakers. In order to simplify the task, our group's project is aimed at implementing Sentiment Recognition in Conversations by analyzing the words and phrases used by the speakers in conversations.

Our task is a classification task which classifies the input sentence into 3 different sentiment classes: negative, neutral, and positive.

The coding can be found in our Github repository:
https://github.com/Meltryllis628/ECR

2 Dataset and Collection

The dataset our group used was the CPED[1] dataset. This dataset is a categorical dataset. Dialogues in the dataset are collected from 40 different Chinese TV shows. This dataset contains 94,187 sentences for the train set, 11,137 sentences for the valid set, and 27,438 sentences for the test set. The sentences are labeled into 3 classes of sentiments and 13 classes of emotions. The data distribution of the original train set is:

	Dialogue_ID	Utterance	Sentiment	Emotion
20	01_004	我说呢这么年轻	neutral	neutral
21	01_005	妈	positive	relaxed
22	01_005	你不要叫我妈	negative	anger
23	01_005	我不是你妈	negative	anger
24	01_005	妈，我错了	negative	depress
25	01_005	你没错	negative	anger
26	01_005	我错了	negative	anger
27	01_005	我为什么要生你	negative	anger
28	01_005	我吃饱了撑的我就不该生你	negative	anger
29	01_005	方一凡我真的不明白你脑子里面天天在想什么	negative	anger
30	01_005	你是不是真的不知道现在高几了	negative	anger

Figure 1: samples of different sentences in dataset

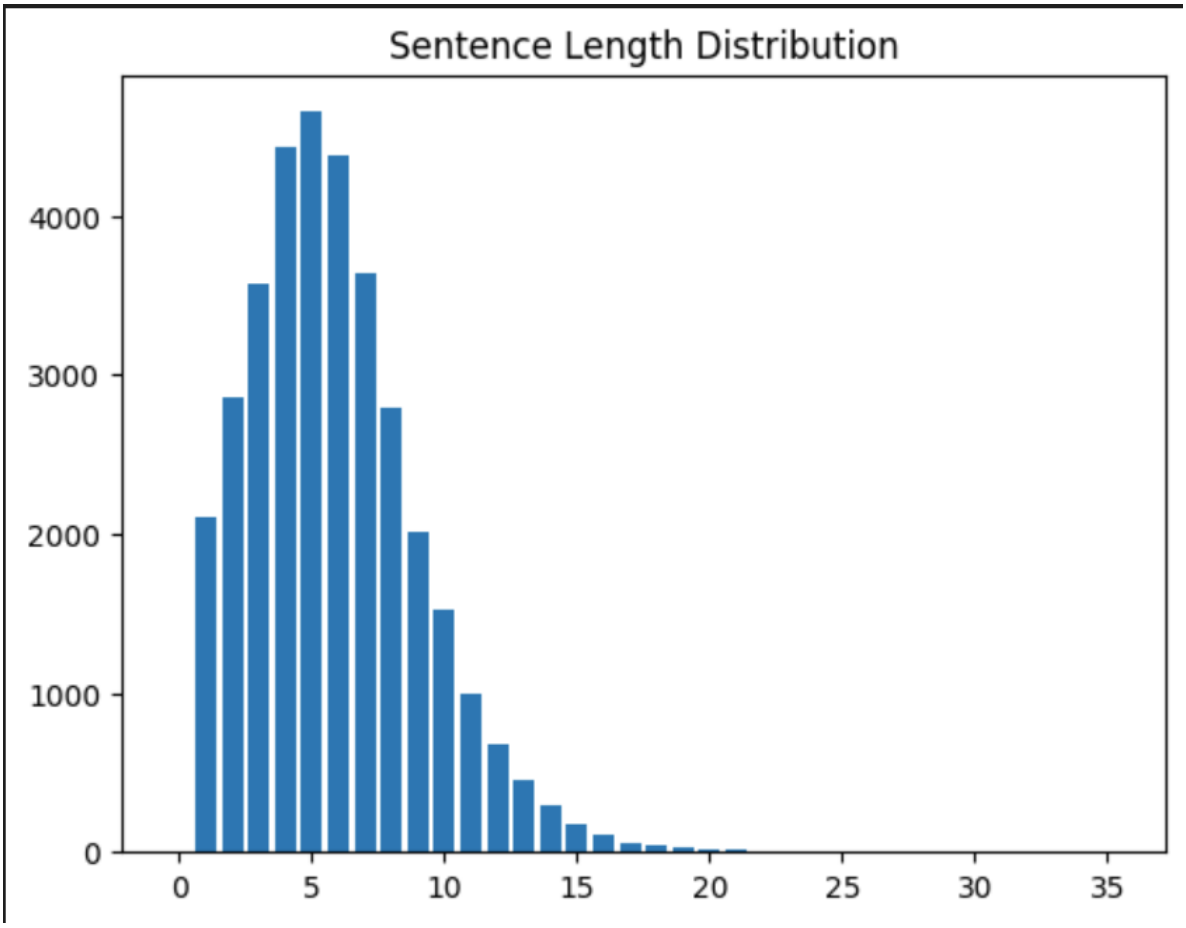

Figure 2: distribution of sentence length in train set

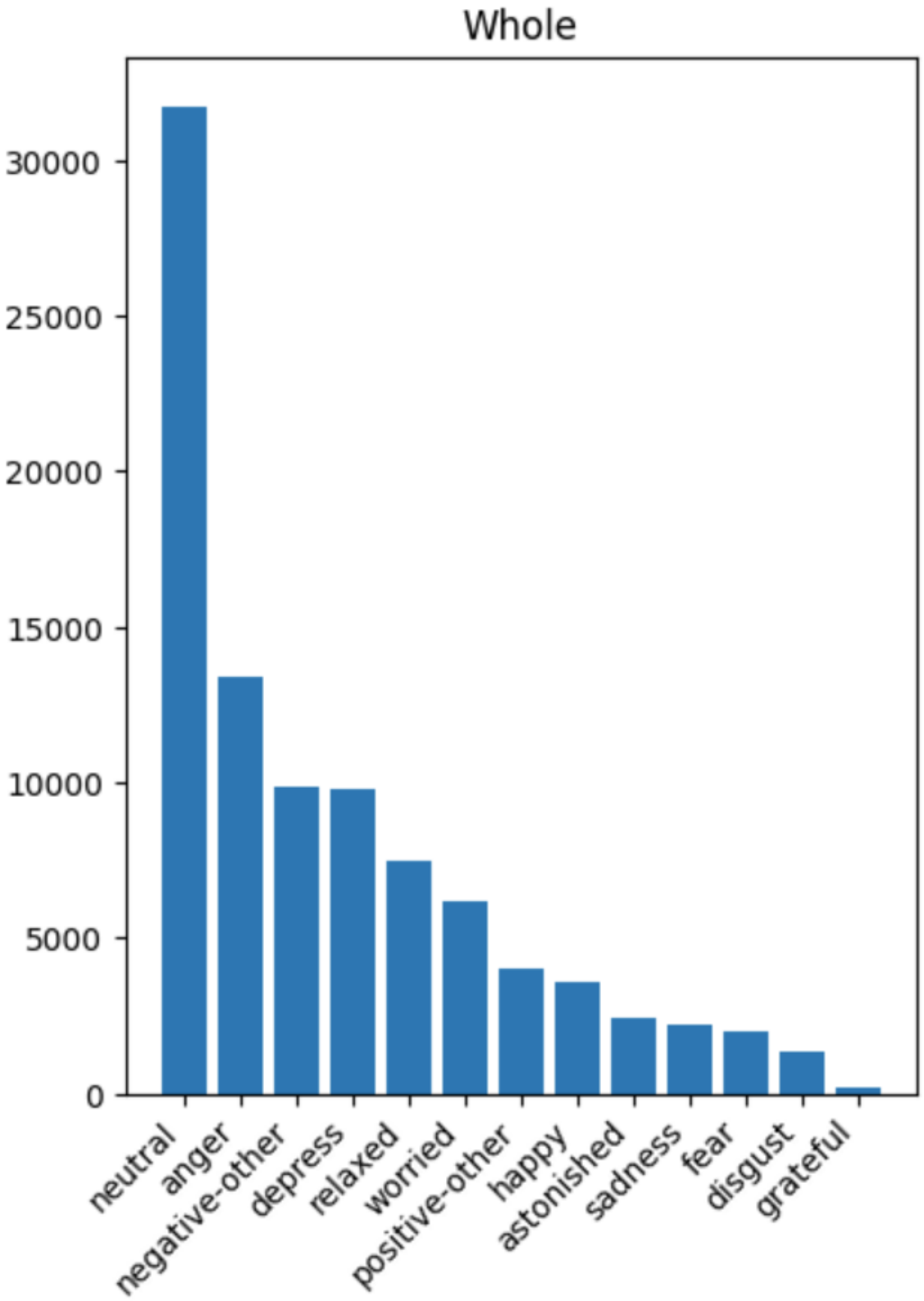

Figure 3: distribution of emotions in train set

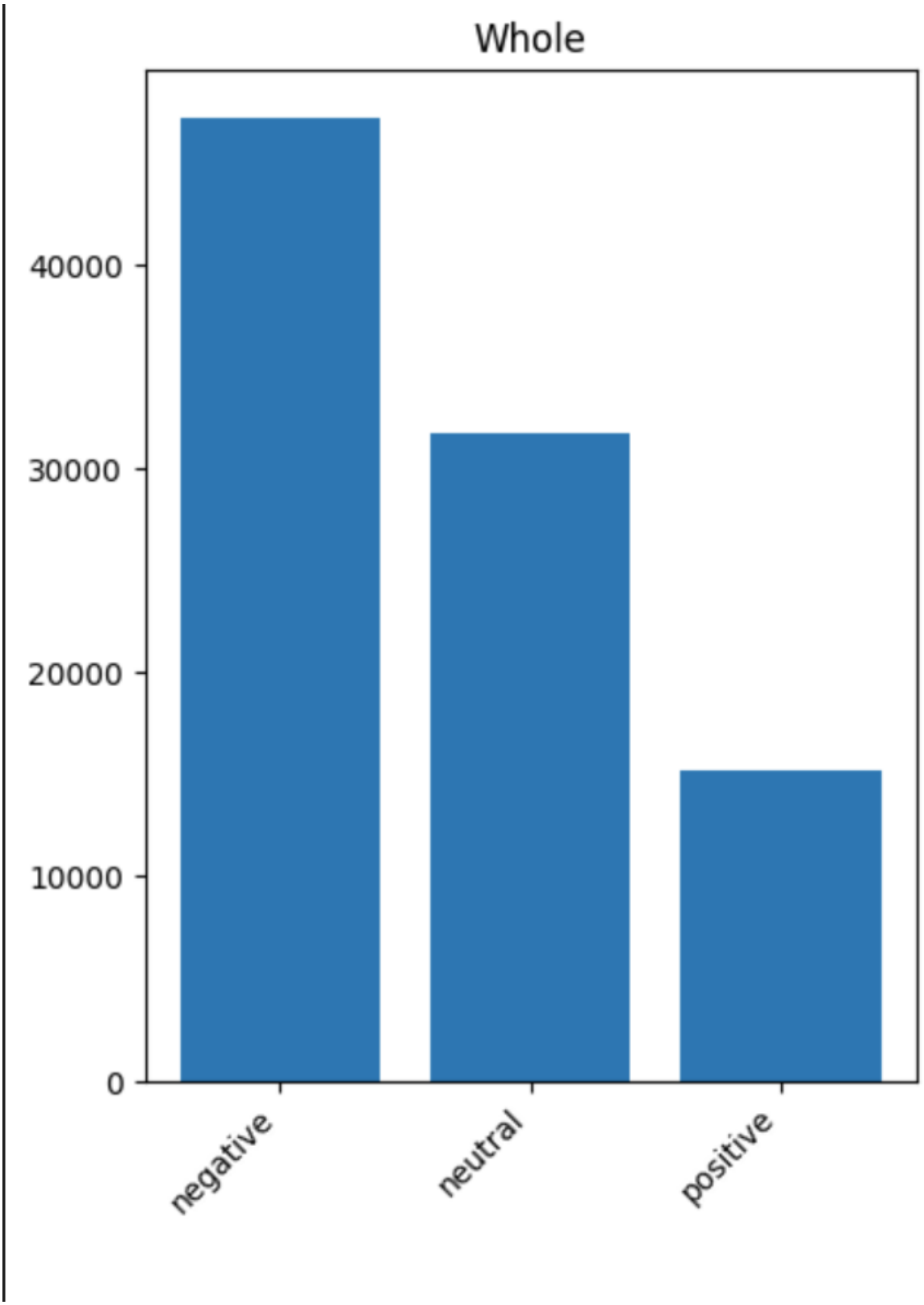

Figure 4: distribution of sentiments in train set

3 Data Pre-processing

In the convenience of embedding and in order to have a better distribution of sentence length, we removed long sentences together with the dialogue they belong to and sentences that contain English. The final data we feed to our model contains 80362 sentences for the train set, 9763 sentences for the valid set, and 24311 sentences for the test set. And in order to simplify the task, we choose to train our model to classify the sentiments of the sentences. The maximum sentence length is 14. The sentiment distribution after processing is:

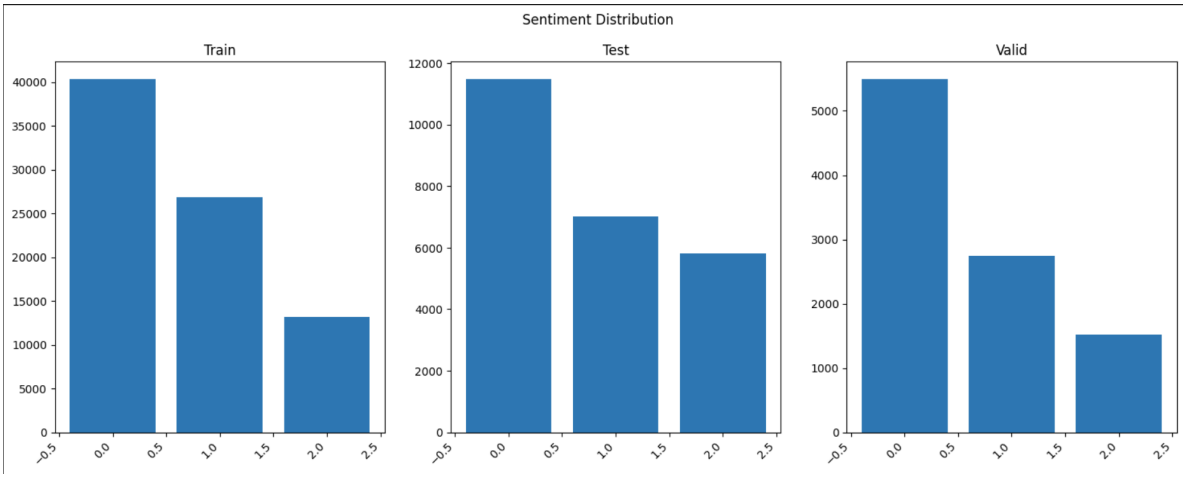

Figure 5: 0:negative 1:neutral 2:positive

For word tokenization, we used the LTP library to split the sentences into word tokens. For word embedding, we tried to train our own embedding but it was much too time-consuming and too large to train. So we decided to use pre-trained word embeddings. We tried fastTest, Chinese-Word-Vectors, and Tencent AI Lab Embedding[2]N18-2028. We finally decided to use Tencent AI Lab Embedding since its embedding dimension is 100 and thus allows us to take the entire dataset into training without running out of GPU RAM. For the person names that appear in the conversation but don't exist in the embedding dictionary, we choose to split it into characters and compute the average value as the embedding. For other word tokens that doesn't exist in the embedding dictionary, we split it into characters and then do the embedding by characters.

4 Problem and Algorithm/Model

4.1 Problem Statement

Our group's project aims at classifying the sentiment of a given sentence in a Chinese dialogue by learning the words and phrases used in Chinese conversations. The original purpose was to recognize the emotions of given sentences. However, since the dataset provides 13 different classes of emotions, which is too complex, we simplified the task into recognizing the sentiment (3 different classes) of given sentences.

4.2 Model Description

The model we choose to use is the bc-LSTM model. The reason for making such a choice is that it is a widely used model for NLP tasks which is relatively easier to implement while having a good performance. The structure of our model is shown below. We take two sentences before the target sentence together with the target sentence as the inputs.

```
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
X_prev Input (InputLayer)       [(None, 15, 100)]    0           []

X Input (InputLayer)            [(None, 15, 100)]    0           []

X_next Input (InputLayer)       [(None, 15, 100)]    0           []

bidirectional_5 (Bidirectional  (None, 512)          731136      ['X_prev Input[0][0]',
)                                                                 'X Input[0][0]',
                                                                  'X_next Input[0][0]']

concatenate_5 (Concatenate)     (None, 1536)         0           ['bidirectional_5[0][0]',
                                                                  'bidirectional_5[1][0]',
                                                                  'bidirectional_5[2][0]']

reshape_5 (Reshape)             (None, 3, 512)       0           ['concatenate_5[0][0]']

LSTM_layer (LSTM)               (None, 300)          975600      ['reshape_5[0][0]']

batch_normalization_5 (BatchNo  (None, 300)          1200        ['LSTM_layer[0][0]']
rmalization)

Output_layer (Dense)            (None, 3)            903         ['batch_normalization_5[0][0]']

==================================================================================================
Total params: 1,708,839
Trainable params: 1,708,239
Non-trainable params: 600
__________________________________________________________________________________________________
```

Figure 6: model summary

```python
def make_model(input_shape,h_n = 100, num_classes=3):
    X_prev_1 = keras.Input(shape=input_shape, name="X_prev Input")
    X_prev_2 = keras.Input(shape=input_shape, name="X Input")
    X = keras.Input(shape=input_shape, name="X_next Input")
    # masked1 = Masking(0)(X_prev_1)
    # masked2 = Masking(0)(X_prev_2)
    # masked3 = Masking(0)(X)
    lstm = Bidirectional(LSTM(256, dropout=0.2, kernel_regularizer=regularizers.L2(0.0001)))

    lstm1 = lstm(X_prev_1)
    lstm2 = lstm(X_prev_2)
    lstm3 = lstm(X)

    inp = Concatenate(axis=-1)([lstm1, lstm2, lstm3])

    inp = Reshape((3, 2*256, )) (inp)
    tmp = LSTM(300, dropout=0.1, recurrent_dropout=0.1,kernel_regularizer=regularizers.L2(0.0001), name="LSTM_layer")(inp)
    bn = BatchNormalization()(tmp)
    outputs = Dense(num_classes, activation="softmax", name="Output_layer")(bn)
    return keras.Model([X_prev_1,X_prev_2,X], outputs)
```

Figure 7: model codes

The early model we used was following the structure of the model given in the instruction[3]. We noticed that it would suffer from overfitting. In order to solve that problem, we tried adding L2 regularizers to the LSTM layers and adding masking layers to the inputs to ignore zero paddings. However, in the training process, we noticed that the masking layers didn't improve the performance a lot but significantly increased the train time. So we decided to drop the masking layers. We also did some hyperparameter tuning to find the best performance of the model.

5 Evaluation Methodology

We evaluated the performance of our model based on the loss and accuracy of the model on the validation set; the confusion matrix of the model on the train set, validation set, and test set; the f1 score of the model on the test set. The loss value is computed using categorical_crossentropy provided in keras. We trained the model multiple times to evaluate its performance. In the most recent training, after training for 25 epochs, we early stopped as the valid loss didn't become better in 10 epochs. The best performance can be found in epoch 19. The valid loss at epoch 19 is 0.9091 and the valid accuracy is 0.5930. The valid loss plot and accuracy plot is:

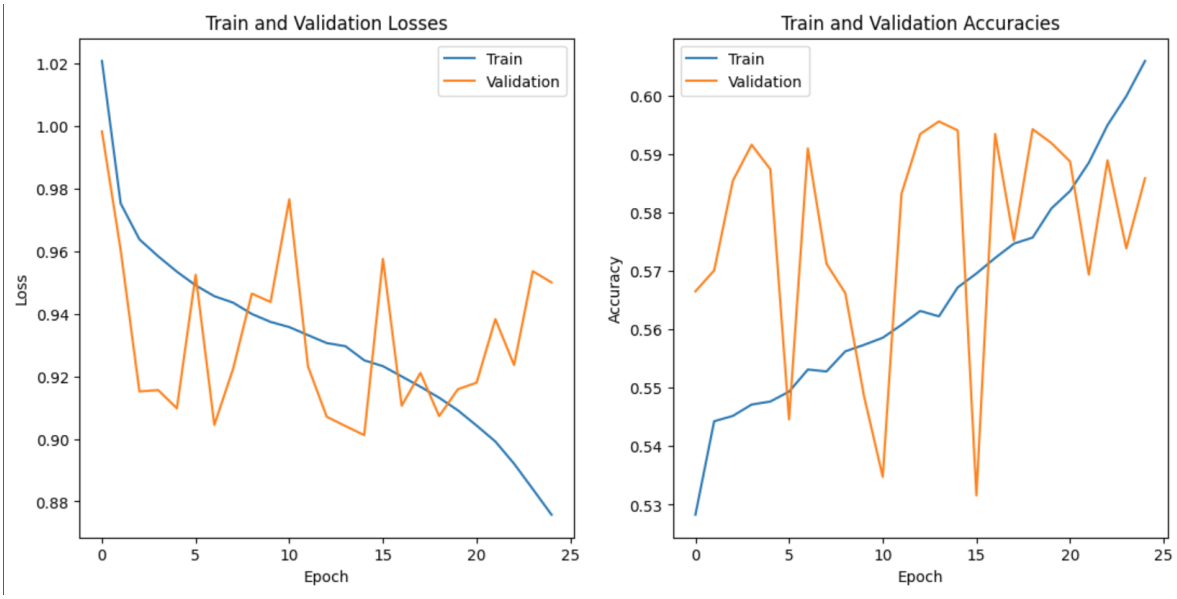

Figure 8: loss and accuracy plot for train set and valid set

The plot without early stopping is shown below. We can observe the overfitting problem more clearly in this plot.

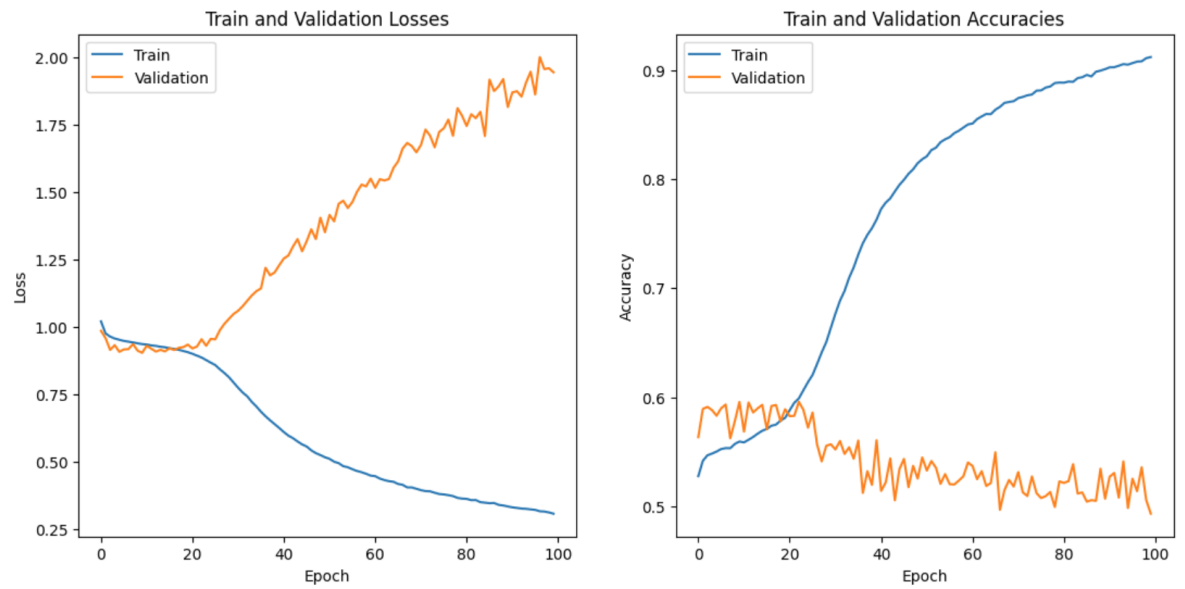

Figure 9: loss and accuracy plot for train set and valid set without early stopping

The confusion matrix for train set, validation set, test set, and f1 score computed on test set are:

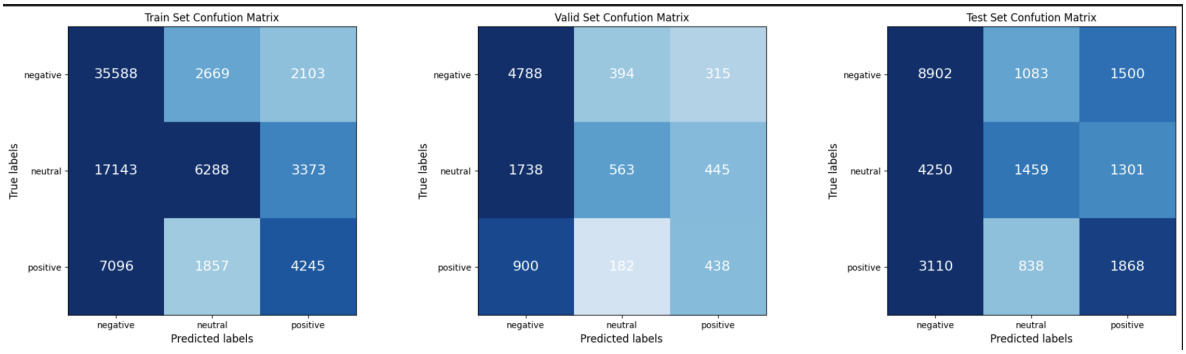

Figure 10: confusion matrix

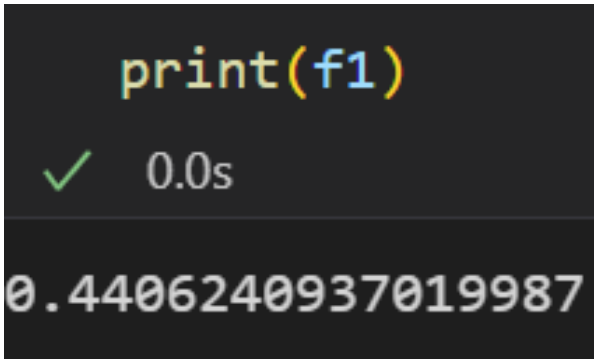

Figure 11: confusion matrix

6 Results and Discussions

As we can observe from the result, the model still suffers from the overfitting problem. We referred to the original paper on the CPED dataset and found that the result is acceptable. That's because the emotion mutation makes the ERC task significantly different from other long text emotion recognition tasks. In order to solve this problem, we need to further study the influence of emotion consistency and emotion mutation on the ERC task. The results and emotion transition figure from the CPED paper are[1]:

TABLE VII
SENTIMENT RECOGNITION IN CPED (NEG.: NEGATIVE, NEU.: NEUTRAL, POS.: POSITIVE AND AVG.: AVERAGE).

Model	Accuracy				Macro-F1
	Neg.	Neu.	Pos.	Avg.	
TextCNN [60]	64.51	24.56	14.04	48.90	34.37
TextRNN [61]	62.69	33.21	15.32	47.89	37.07
TextRCNN [62]	64.04	31.03	18.78	49.13	37.95
FastText [63]	**65.30**	24.76	0.95	48.62	30.33
BERT [58]	59.97	**42.98**	32.60	48.96	45.18
bcLSTM [64]	59.06	38.86	28.28	49.65	45.40
DialogueRNN [65]	59.06	36.16	**44.97**	48.57	44.11
DialogueGCN [66]	60.65	36.99	40.19	47.69	45.12
EmoBERTa [68]	59.24	35.02	41.53	48.09	44.60
DialogXL [67]	60.45	41.45	41.74	51.24	46.96
BERT+AVG+MLP	61.40	40.10	42.95	**51.50**	**48.02**

Figure 12: confusion matrix

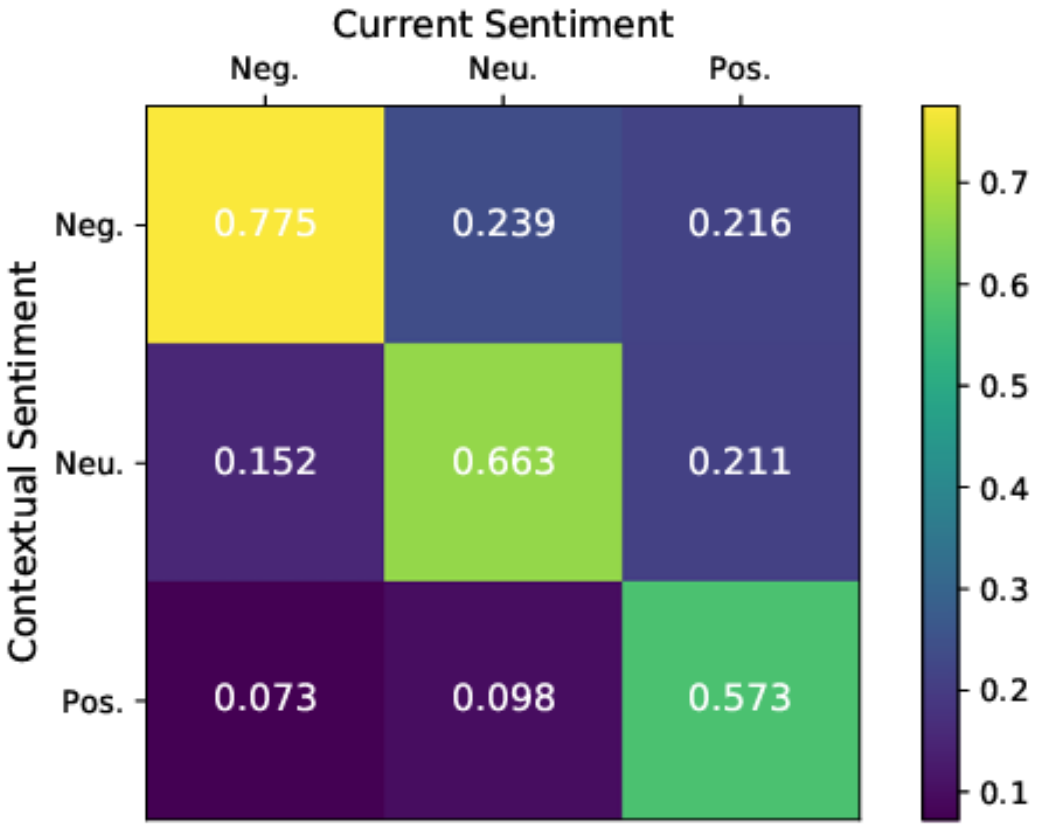

Figure 13: confusion matrix

References

[1] Yirong Chen, Weiquan Fan, Xiaofen Xing, Jianxin Pang, Minlie Huang, Wenjing Han, Qianfeng Tie, and Xiangmin Xu. CPED: A large-scale chinese personalized and emotional dialogue dataset for conversational ai. *arXiv preprint arXiv:2205.14727*, 2022.

[2] Yan Song, Shuming Shi, Jing Li, and Haisong Zhang. Directional skip-gram: Explicitly distinguishing left and right context for word embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 175–180. Association for Computational Linguistics, 2018.

[3] Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Navonil Majumder, Amir Zadeh, and Louis-Philippe Morency. Context-dependent sentiment analysis in user-generated videos. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 873–883, Vancouver, Canada, July 2017. Association for Computational Linguistics.