Assignment 3 Emotion Recognition

Data

In this task, we worked on the Twitter Emotion Recognition task by using the Tweeteval dataset to train a CNN model.

We performed the following preprocessing operations:

- Filtering Line breaks
- Filtering User names
- Turning tags into texts
  - We noticed that there are two kinds of tags: In-sentence or After-sentence. For the former, we presume deleting these tags will cause chaos in sentence meanings. So we turned all tags into words.
- Stemming
- Turn emoji into text
  - Emojis can play important roles in expressing emotions. So, instead of deleting them directly (like what we did in exercise 1 and 2), we turned them into words using demojize().
- Filtering Symbols and nums

Model Training

We picked **Joy** and **Anger** as the class tags and generated dataset 1. We filtered the labels and the corresponding texts and relabeled them in the class TextLoader. Then, we did data cleaning in the class TweetCleaner and generated their torch DataLoader objects with the class ContextGenerator.

We picked **Joy** and **Sadness** as another group of tags and generated dataset 2 with the same operation as dataset 1.

We trained and tuned model 1 on dataset 1 with the following parameters.

optimizer	batch sizes	dropout	filter sizes	acc_pos	acc_neg	F1
Adadelta	4	0.2	[3, 4, 5]	68.8%	26.8%	0.30
Adadelta	**8**	**0.2**	**[3, 4, 5]**	**87.5%**	**17.5%**	**0.25**
Adadelta	**4**	**0.5**	**[3, 4, 5]**	**65.6%**	**51.5%**	**0.49**
Adadelta	8	0.5	[3, 4, 5]	80.6%	14.4%	0.19
Adam	4	0.2	[3, 4, 5]	6.9%	92.8%	0.53
Adam	8	0.2	[3, 4, 5]	4.4%	94.8%	0.53
Adam	4	0.5	[3, 4, 5]	10.0%	82.5%	0.49
Adam	8	0.5	[3, 4, 5]	5.0%	96.9%	0.54
Adadelta	4	0.5	[3, 5, 7]	72.5%	32.0%	0.36
Adadelta	**2**	**0.5**	**[3, 5, 7]**	**35.6%**	**69.1%**	**0.50**
Adadelta	3	0.5	[3, 5, 7]	61.2%	47.4%	0.44
Adadelta	4	0.3	[3, 4, 5]	63.8%	26.8%	0.28
Adadelta	4	0.1	[3, 4, 5]	48.8%	51.5%	0.43
Adadelta	4	0.4	[3, 4, 5]	4.4%	87.6%	0.50

From the results above, we got the three best hyperparameter settings. From the 3 best parameters groups, we find the Adadelta optimizer is better than Adam. Since Adadelta is very fast and converges rapidly, we can infer that the epochs are not enough for an Adam optimizer to converge. We also discovered that when the batch size is bigger, in order to get better results, the dropout rate should be lower, and the filter size should be smaller. We think this is because there is a balance in the 'information amount' that a model learns to perform better.

We then trained model 2 on dataset 2 with the 3 parameter groups. Finally, we made predictions on the test-set of both datasets using model 2 and got the following results.

- **Joy-Anger Dataset**

optimizer	batch sizes	dropout	filter sizes	acc_pos	acc_neg	F1
Adadelta	8	0.2	[3, 4, 5]	16.5%	89.7%	0.48
Adadelta	4	0.5	[3, 4, 5]	11.5%	92.2%	0.56
Adadelta	2	0.5	[3, 5, 7]	20.4%	87.2%	0.56

- **Joy-Sadness Dataset**

optimizer	batch sizes	dropout	filter sizes	acc_pos	acc_neg	F1
Adadelta	8	0.2	[3, 4, 5]	7.6%	91.3%	0.69
Adadelta	4	0.5	[3, 4, 5]	3.9%	91.1%	0.63
Adadelta	2	0.5	[3, 5, 7]	9.7%	86.3	0.61

We noticed that on both datasets, the accuracy scores are unbalanced. For one tag, it gets a high accuracy, but for another, the result is the opposite. We believe this is because of the unbalanced distribution of labels in the test dataset. This situation occurs when the number of data with one label is much greater than another. The solution is to enlarge the dataset and 'generate' more samples for the tag with fewer data. Another guess is that there is overfitting, but since the dropout rate is 0.5, we would reduce the possibility of this situation. We also found that for model 2, the F1 score is larger on Joy-Sadness Dataset than on the Joy-Anger Dataset. This slight difference shows the CNN model performance on its familiar dataset and the dataset from a new domain.