DM-2024-Lab2

Yang, Hsueh

1 Load Data

1.1 Preparing Data

1.1.1 Download the dataset, including the following files:

dataset/
|-- data_indentification.csv
|-- emotion.csv
|-- sampleSubmission.csv
\-- tweets_DM.json

Description of each file:

- data_indentification.csv: Assign each **tweet_id** to a train or test label.
- Emtion.csv: Assign each **tweet_id** in an emotion label.
- sampleSubmission.csx: Demonstration of format of submission.csv.
- tweets_DM.json: Primary dataset, containing tweets.

1.1.2 Load tweets_DM.json into a dictionary, then take out the portion of "tweet" into a DataFrame.

1.1.3 According to data_identification.csv to distinguish which "tweet_id" belongs to train or test dataset, and according to emotion.csv to distinguish which "tweet_id" belongs to which emotion label. Finally, transform each emotion label into numerical ones using one-hot encoding, and extracted data shown in below.

	hashtags	tweet_id	text	emotion	label
0	[Snapchat]	0x376b20	People who post "add me on #Snapchat" must be ...	anticipation	1
1	[freepress, TrumpLegacy, CNN]	0x2d5350	@brianklaas As we see, Trump is dangerous to #...	sadness	5
2	[]	0x1cd5b0	Now ISSA is stalking Tasha 😂😂😂 <LH>	fear	3
3	[authentic, LaughOutLoud]	0x1d755c	@RISKshow @TheKevinAllison Thx for the BEST TI...	joy	4
4	[]	0x2c91a8	Still waiting on those supplies Liscus. <LH>	anticipation	1
5	[]	0x368e95	Love knows no gender. 😢😭 <LH>	joy	4

1.2 Data Observation

Following I will introduce the observation of data and the most effective preprocessing process in my homework. Following contents are discussed with my classmates and I conclude it in here.

1.2.1 According to the text in the block below, we find out that the <LH> is as same as

<mask> token in LM, such as RoBERTa. Therefore, we replace it into the special token that used LM can recognize it as a mask token.

- Example:
  People who post "add me on #Snapchat" must be dehydrated. Cuz man.... that\'s <LH>

- Becomes:
  People who post "add me on #Snapchat" must be dehydrated. Cuz man.... that\'s <mask>

1.2.2 We observe that there are some data containing too much <LH> in their tweets, the counting shows in the third and fourth block below. I thought if there are too many LH, @, and # may be meaningless or hard to distinguish, so I remove some data exceed threshold.

- Example:

```
{
    "tweet_id": "0x2aaa9d",
    "tweets": "@berksmike @SussexMistress @bradfordmistres @MistressYvonne @domstrapon
@MsScarlettBlack @missjulia2013 @MadameC5 Really <LH> 😊"
}
```

2 Data Preprocessing

2.1 Find the number of <LH> or portion of <LH> which is larger than 4 or larger 2/3.

```
noise_rate = 2 / 3
lh_max = 4

lh_rate_count = train_df[train_df["noise rate"] >= noise_rate].shape[0]
lh_count = train_df[train_df["<LH>"] > lh_max].shape[0]

print(f"Number of rows with (noise rate >= {noise_rate:.2f}): {lh_rate_count}")
print(f"Number of rows with (<LH> > {lh_max}): {lh_count}")


# Filter the rows with <LH> rate >= 2/3 or <LH> >= 5
filtered_df = train_df[
    (train_df["noise rate"] >= noise_rate) | (train_df["<LH>"] > lh_max)
]

# Convert the filtered dataframe to a list of dictionaries
filtered_texts = filtered_df.to_dict(orient="records")

# Save the list of dictionaries to a JSON file
with open("filtered_texts.json", "w") as outfile:
    json.dump(filtered_texts, outfile, ensure_ascii=False, indent=4)
```

```
Number of rows with (noise rate >= 0.67): 38505
Number of rows with (<LH> > 4): 39817
```

2.2 Remove them.

```
# Remove rows with noise rate >= 2/3 or <LH> > 4
train_df = train_df[
    (train_df["noise rate"] < noise_rate) & (train_df["<LH>"] <= lh_max)
]
```

```
print("train data")
lh_rate_count = train_df[train_df["noise rate"] >= noise_rate].shape[0]
lh_count = train_df[train_df["<LH>"] > lh_max].shape[0]

print(f"Number of rows with (noise rate >= {noise_rate:.2f}): {lh_rate_count}")
print(f"Number of rows with (<LH> > {lh_max}): {lh_count}")
```

```
train data
Number of rows with (noise rate >= 0.67): 0
Number of rows with (<LH> > 4): 0
```

2.3 Preprocessing Function

Briefly preprocessing with some effective process.

```
import re
from spellchecker import SpellChecker

spell = SpellChecker()


def preprocess_tweet(text):
    text = re.sub(r"(https?://)?[\w.-]+\.com(\.\w+)?", "<URL>", text)  # replace URLs
    text = re.sub(r"<LH>", "<mask>", text)  # replace "<LH>" with "<mask>"
    text = re.sub(r"\s+", " ", text)  # remove extra whitespaces

    return text
```

3 Model Training

3.1 In this part, I used a pre-trained encoder model from Twitter, Twitter/twhin-bert-base, to do a classification task. The hyperparameters are as follows:

```
train_batch_size = 256
val_batch_size = 256
dropout_rate = 0.1
lr = 2e-5
epochs = 5
val_split = 0.1
```

3.2 Construct the required pre-trained model and define the downstream task to perform the classification task, and setup the optimizer and loss function.

```
model_name = "Twitter/twhin-bert-base"
```

- model definition

```python
class TweetEmotionClassifier(torch.nn.Module):
    def __init__(self, model_name, dropout=0.1):
        super().__init__()
        self.bert = AutoModel.from_pretrained(model_name)
        self.dropout = torch.nn.Dropout(p=dropout)
        self.linear = torch.nn.Linear(self.bert.config.hidden_size, 8)

    def forward(self, **kwargs):
        output = self.bert(**kwargs)
        cls_output = output.last_hidden_state[:, 0, :]
        cls_output = self.dropout(cls_output)
        logits = self.linear(cls_output)

        return logits

    def extract_features(self, **kwargs):
        output = self.bert(**kwargs)
        cls_output = output.last_hidden_state[:, 0, :]
        return cls_output


model = TweetEmotionClassifier(model_name, dropout=dropout_rate)
```

- optimizer and loss function

```python
from torch.optim import AdamW
from torchmetrics.classification import MulticlassAccuracy, MulticlassF1Score

optimizer = AdamW(model.parameters(), lr=lr)

criteria = torch.nn.CrossEntropyLoss()

acc = MulticlassAccuracy(num_classes=8).to(device)
f1 = MulticlassF1Score(num_classes=8).to(device)
```

4 Results

4.1 Metric Score of Validation

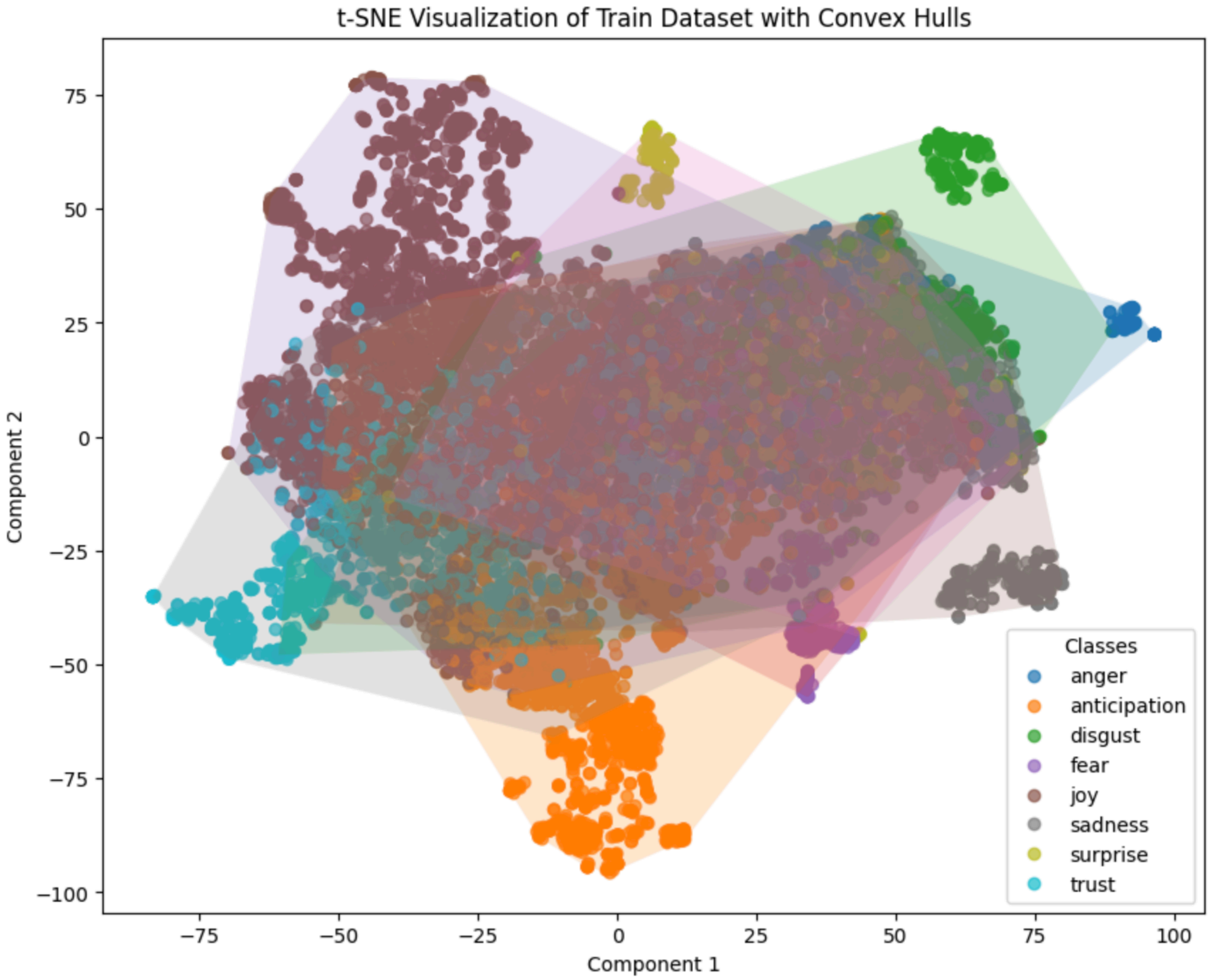
```
Training Epoch [5/5]: 100%|██████████| 4838/4838 [44:19<00:00,  1.82it/s, loss=0.703, lr=2e-5]
Validation Epoch [5/5]: 100%|██████████| 538/538 [01:43<00:00,  5.17it/s, loss=0.958]
Accuracy: 0.5944
F1 Score: 0.6053
```

4.2 LB

13	▾ 1	Yang Hsueh	0.54842	15	1d

4.3 t-SNE:

Although there are some clusters easy to observe around the graph, but in the middle, it is still chaos.


t-SNE Visualization of Train Dataset with Convex Hulls
Component 2
Component 1
75
50
25
0
−25
−50
−75
−100
−75
−50
−25
0
25
50
75
100
Classes
anger
anticipation
disgust
fear
joy
sadness
surprise
trust