DM 2024 Lab2 report

Data preprocessing

- 讀取data identification資料，將資料切分成train與test資料

```
input_path = '/kaggle/input/dm-2024-isa-5810-lab-2-homewor
k'
# Read train and test identification data
import pandas as pd
data_ids = pd.read_csv(f'{input_path}/data_identification.
csv')
print('data_identification')
print(data_ids.head())

# Split the data into training and testing sets.
train_data_ids = data_ids[data_ids['identification'] == 't
rain'].drop(['identification'], axis=1)
test_data_ids = data_ids[data_ids['identification'] == 'te
st'].drop(['identification'], axis=1)
print('train')
print(train_data_ids.head())
```

- 讀取emotion.csv中情緒類別資訊

```
# read
emotion_labels = pd.read_csv(f'{input_path}/emotion.csv')
emotion_labels.head()
```

- 讀取tweets_DM.json資料，並且將其格式處理成表格模式，將tweets的文本資料與訓練資料和測試資料合併，並將emotion的label放入訓練資料的欄位

```
tweets_df = pd.read_json(f'{input_path}/tweets_DM.json', l
ines=True)
```

```
print(tweets_df.head())

source_df = pd.json_normalize(tweets_df['_source'])

tweets_df = pd.concat([tweets_df.drop(columns=['_sourc
e']), source_df], axis=1)

print(tweets_df.head())

train_data = pd.merge(train_data_ids, emotion_labels, on
='tweet_id', how='inner')
train_data = pd.merge(train_data, tweets_df, left_on='twee
t_id', right_on='tweet.tweet_id', how='inner')
train_data.head()

test_data = pd.merge(test_data_ids, tweets_df, left_on='tw
eet_id', right_on='tweet.tweet_id', how='inner')
test_data.head()
```

- 去除資料中不必要的欄位，並重新命名欄位名稱

```
# drop useless column
drop_columns = ['_score','_index','_crawldate','_type','tw
eet.hashtags','tweet.tweet_id']
train_data.drop(drop_columns,axis=1,inplace=True)
test_data.drop(drop_columns,axis=1,inplace=True)
# rename column name
train_data = train_data.rename(columns={"tweet.text": "tex
t"})
test_data = test_data.rename(columns={"tweet.text": "tex
t"})
```

[8]:
```
train_data.head()
```

[8]:

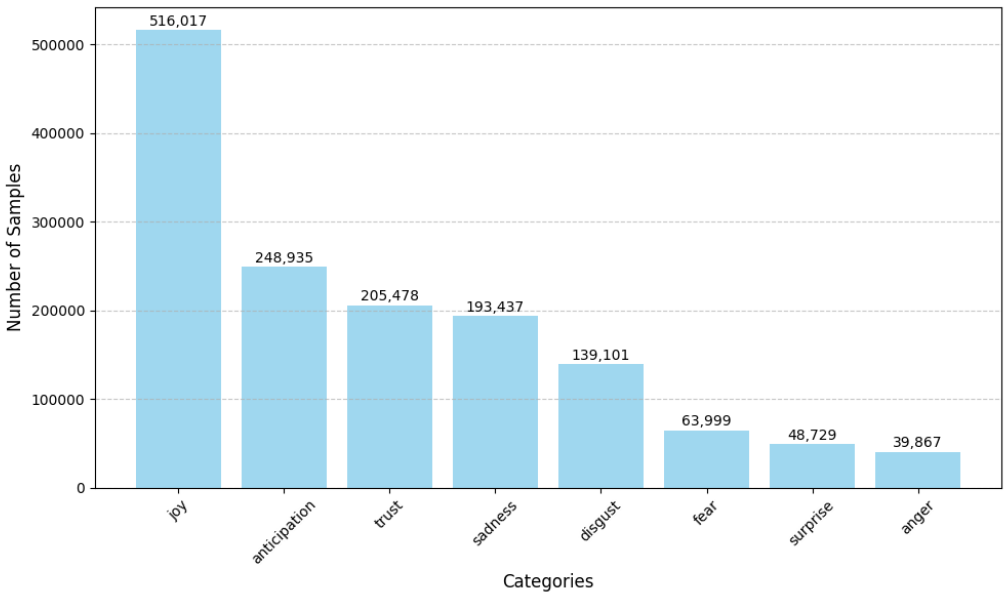
	tweet_id	emotion	text
0	0x29e452	joy	Huge Respect @JohnnyVegasReal talking about l...
1	0x2b3819	joy	Yoooo we hit all our monthly goals with the ne...
2	0x2a2acc	trust	@KIDSNTS @PICU_BCH @uhbcomms @BWCHBoss Well do...
3	0x2a8830	joy	Come join @ambushman27 on #PUBG while he striv...
4	0x20b21d	anticipation	@fanshixieen2014 Blessings!My #strength little...

[9]:
```
test_data.head()
```

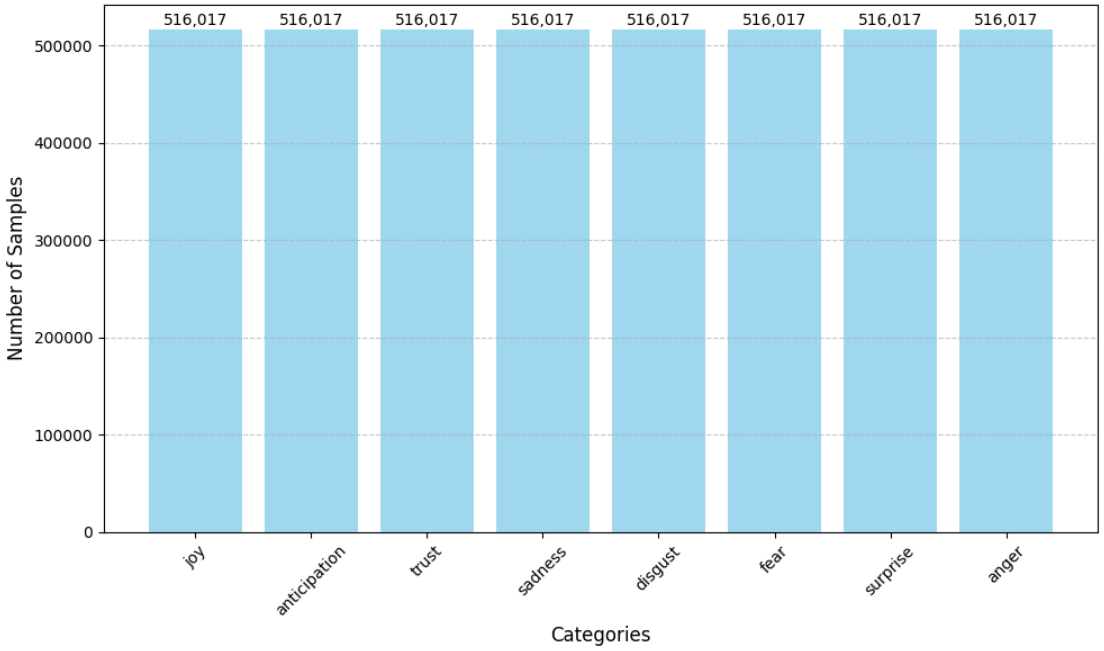
[9]:

	tweet_id	text
0	0x28cc61	@Habbo I've seen two separate colours of the e...
1	0x2db41f	@FoxNews @KellyannePolls No serious self respe...
2	0x2466f6	Looking for a new car, and it says 1 lady owne...
3	0x23f9e9	@cineworld "only the brave" just out and fount...
4	0x1fb4e1	Felt like total dog 💩 going into open gym and ...

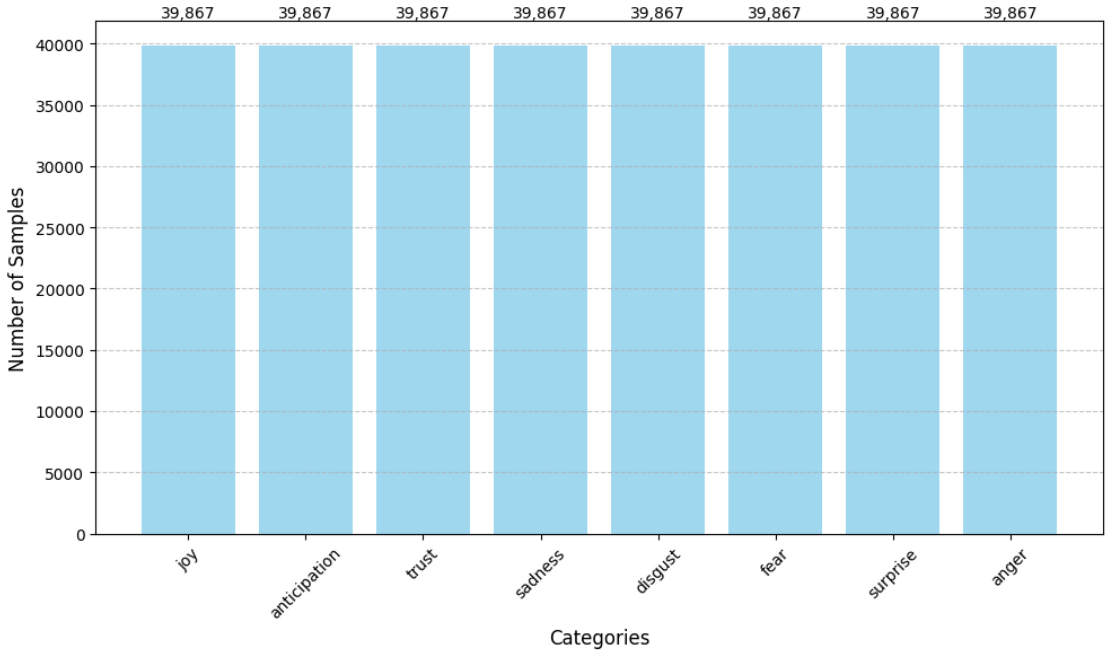
- 資料類別分布的分析
  - 當資料內數量比例超過1:4時，建議在分析前將資料不平衡的問題納入考量 (https://ithelp.ithome.com.tw/m/articles/10294614)
  - 原始訓練資料集中類別資料分布



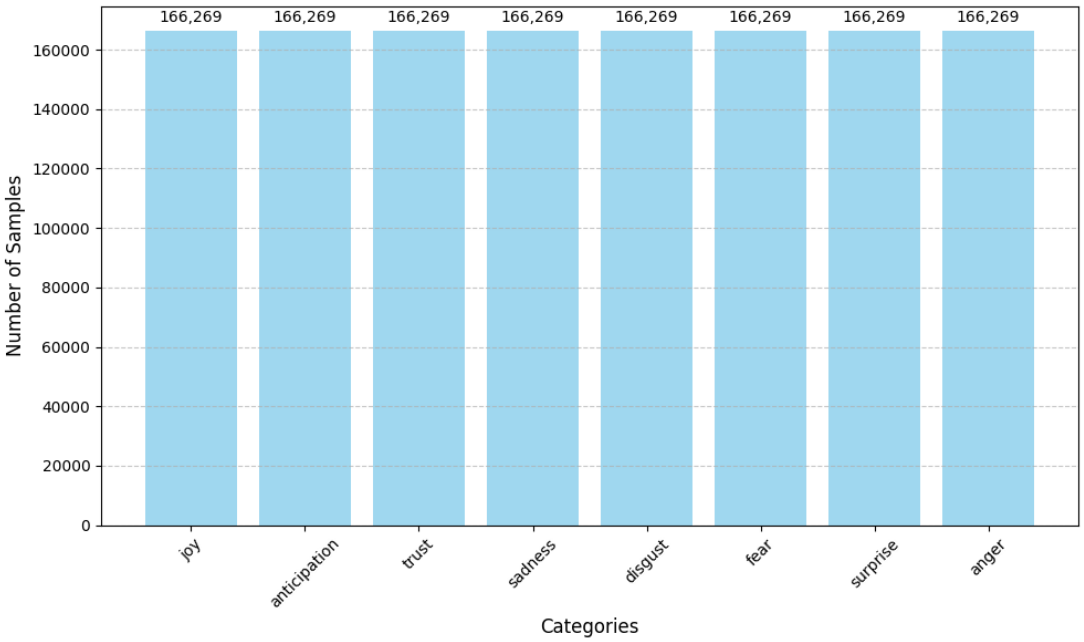
  - 上採樣後的類別資料分布



- 下採樣後的類別資料分布



- 中位數採樣後的類別資料分布
  - 想法: 取得類別資料的中位數做為採樣目標數，少於中位數的類別進行上採樣，大於中位數的類別進行下採樣



- 資料缺失值的分析
  - 未檢查出有缺失值資料

```
# check whether there is null data
train_data["text"].isna().sum()
```

- 將emotion的字串標籤轉換成整數型態的標籤，以利於後續模型訓練

```
# label encode the emotion data
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
train_data['labels'] = labelencoder.fit_transform(train_data
```

- 前處理文本資料
  - tokenize方法
    - 方法1: 使用nltk中的TweetTokenizer對text進行tokenize
    - 方法2: 使用bertweet預設tokenizer，可以參考 https://github.com/VinAIResearch/BERTweet中的介紹
  - 處理text資料

- 我參考了https://github.com/VinAIResearch/BERTweet中TweetNormalizer的程式，他會對於一些hashtag標籤以及http url網址統一成一個可識別的token，並且會處理text中縮寫不一致的問題。

```
from transformers import AutoTokenizer
from nltk.tokenize import TweetTokenizer
model_name = "vinai/bertweet-base"
# tokenize method 1: nltk TweetTokenizer
tokenizer = TweetTokenizer()
# tokenize method 2: bertweet default tokenizer
tokenizer = AutoTokenizer.from_pretrained(model_name)

def normalizeToken(token):
    lowercased_token = token.lower()
    if token.startswith("@"):
        return "@USER"
    elif lowercased_token.startswith("http") or lowercased_to
        return "HTTPURL"
    elif len(token) == 1:
        return demojize(token)
    else:
        if token == "’":
            return "'"
        elif token == "…":
            return "..."
        else:
            return token


def normalizeTweet(tweet):
    tokens = tokenizer.tokenize(tweet.replace("’", "'").repla
    normTweet = " ".join([normalizeToken(token) for token in

    normTweet = (
        normTweet.replace("cannot ", "can not ")
```

```
        .replace("n't ", " n't ")
        .replace("n 't ", " n't ")
        .replace("ca n't", "can't")
        .replace("ai n't", "ain't")
    )
    normTweet = (
        normTweet.replace("'m ", " 'm ")
        .replace("'re ", " 're ")
        .replace("'s ", " 's ")
        .replace("'ll ", " 'll ")
        .replace("'d ", " 'd ")
        .replace("'ve ", " 've ")
    )
    normTweet = (
        normTweet.replace(" p . m .", "  p.m.")
        .replace(" p . m ", " p.m ")
        .replace(" a . m .", " a.m.")
        .replace(" a . m ", " a.m ")
    )

    return " ".join(normTweet.split())
```

- 將資料打包成Dataset格式，以便後續模型訓練

```
class TweetDataset(Dataset):
    def __init__(self, tweets, labels=None):
        self.tweets = tweets
        self.labels = labels

    def __len__(self):
        return len(self.tweets)

    def __getitem__(self, idx):
        tweet = normalizeTweet(self.tweets[idx])
        encoding = tokenizer(tweet, padding="max_length",
```

```
truncation=True, max_length=128, return_tensors="pt")
        if self.labels is not None:
            return {
                "input_ids": encoding["input_ids"].squeeze
(),
                "attention_mask": encoding["attention_mas
k"].squeeze(),
                "labels": torch.tensor(self.labels[idx], d
type=torch.long),
            }
        else:
            return {
                "input_ids": encoding["input_ids"].squeeze
(),
                "attention_mask": encoding["attention_mas
k"].squeeze(),
            }

train_dataset = TweetDataset(train_split['text'], train_sp
lit['labels'])
eval_dataset = TweetDataset(eval_split['text'], eval_split
['labels'])
```

Model

- 我使用hugging face中的Bertweet model作為pretrained model，並且使用訓練資料進行fine-tuning
    - 參考資料: https://huggingface.co/vinai/bertweet-base
    - 其中BERTweet 是首個針對英語推文（Tweets）進行大規模預訓練的公開語言模型。BERTweet 是基於 RoBERTa 預訓練過程進行訓練的。用於預訓練 BERTweet 的語料庫包含了 8.5 億條英語推文（約 160 億個詞元，大小約 80GB），其中包括 8.45 億條從 2012 年 1 月到 2019 年 8 月期間流式獲取的推文，以及 500 萬條與 COVID-19 大流行相關的推文。

```
from transformers import AutoModel, AutoTokenizer, AutoModelF
orSequenceClassification, Trainer, TrainingArguments
#read pretrained bertweet model
model_name = "vinai/bertweet-base"
model = AutoModelForSequenceClassification.from_pretrained(mo
del_name, num_labels=8)

# Train arguments setting
training_args = TrainingArguments(
    output_dir="./results",
    run_name="bert-finetuning",
    #eval_strategy="no",
    save_strategy="epoch",
    eval_strategy="epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=32,
    num_train_epochs=1,
    weight_decay=0.01,
    logging_dir="./logs",
    report_to=[],
    fp16=True,
    #no_cuda=False,
)

# Define trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=eval_dataset,
    compute_metrics=compute_metrics,
)

# bertweet model fine-tuning
trainer.train()
```

```
# save bertweet-finetuned model
model.save_pretrained("./bertweet-finetuned")
```

Experiment Result

實驗一: 使用nltk中的TweetTokenizer

Public score	Private score
0.53970	0.52692

實驗二: 使用bertweet-base model預設的tokenizer

Public score	Private score
0.57060	0.55577

實驗三: 處理類別資料不平衡問題

- 上採樣
  - 超時，無法執行
- 下採樣

Public score	Private score
0.49259	0.48050

- 中位數採樣

Public score	Private score
0.52621	0.51354