Lab2 Report

Import data

```
import json
import pandas as pd
import numpy as np
import nltk
import matplotlib.pyplot as plt
from collections import Counter

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
```

```
data = []
with open('/kaggle/input/dm-2024-isa-5810-lab-2-homework/tweets_DM.json', 'r') as f:
    for line in f:
        data.append(json.loads(line))

f.close()
```

```
emotion = pd.read_csv('/kaggle/input/dm-2024-isa-5810-lab-2-homework/emotion.csv')
data_identification = pd.read_csv('/kaggle/input/dm-2024-isa-5810-lab-2-homework/data_identification.csv')
```

Use data from json to built a dataframe, and sort it by `identification`

```
df = pd.DataFrame(data)
_source = df['_source'].apply(lambda x: x['tweet'])
df = pd.DataFrame({
    'tweet_id': _source.apply(lambda x: x['tweet_id']),
    'hashtags': _source.apply(lambda x: x['hashtags']),
    'text': _source.apply(lambda x: x['text']),
})
df = df.merge(data_identification, on='tweet_id', how='left')

train_data = df[df['identification'] == 'train']
test_data = df[df['identification'] == 'test']
```

```
train_data = train_data.merge(emotion, on='tweet_id', how='left')
```

```
train_data.drop_duplicates(subset=['text'], keep=False, inplace=True)
```

Sampling

I choose **30% random data** from the training set cause **it is the largest sample the memory can run**.

```
train_data_sample = train_data.sample(frac=0.3, random_state=42)
```

```
y_train_data = train_data_sample['emotion']
X_train_data = train_data_sample.drop(['tweet_id', 'emotion', 'identification', 'hashtags'], axis=1)
train_data_sample
```

	tweet_id	hashtags	text	identification	emotion
861298	0x29573d	[]	@SuicideGirls I will feel high anywhere with t...	train	joy
1162037	0x33a0d6	[Unconscionable]	@SenBobCorker I am a Democrat & I applaud your...	train	disgust
127464	0x22f649	[ReignCane]	Tulsa showing me some love! <LH> #ReignCane @C...	train	trust
1216259	0x207fd5	[]	@CNN Why now? Why not when obama was president...	train	surprise
630563	0x34d404	[]	@jtonerv @rajivj @a2zUserGroup Is that a cardb...	train	joy
...	...	...	...	...	...
1216605	0x2fecef	[teamangie]	Finding it diddicult to watch Kerbers matches ...	train	sadness
241188	0x20f525	[Life]	73 The moments in your life are only once #Lif...	train	anticipation
273638	0x1f9535	[]	@P5HBrazil @FifthHarmony B I vote for @FifthHa...	train	sadness
488366	0x3707ec	[texas, nyc]	Hate will never win. <LH> #texas #nyc	train	joy
1281567	0x2cfb63	[]	Today's travel playlist includes @zoella 's "S...	train	joy

434755 rows × 5 columns

Processing test-train data

Split the test-train set

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_train_data, y_train_data, test_size=0.2, random_state=42, stratify=y_train_data)
```

Use TF-IDF Vectorizer to transform the data. This place I choose `max_features=2000`. The reason is also the memory, 2000 is the biggest number that can run in the memory, or it may crash.

```
tfidf = TfidfVectorizer(max_features=2000)
X = tfidf.fit_transform(X_train['text']).toarray()
X_test = tfidf.transform(X_test['text'])
```

Encode the emootion label, make sure the label can be calculated.

```
le = LabelEncoder()
y = le.fit_transform(y_train)
y_test = le.transform(y_test)
```

Random Forest Classifier

```
classifier = RandomForestClassifier()
classifier.fit(X, y)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
y_pred = classifier.predict(X_test)
```

Compare the test and the test prediction. Preliminarily watch the accuracy after the above steps.

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

```
0.5144276661568009
```

```
X_test_data = test_data.drop(['tweet_id', 'identification', 'hashtags'], axis=1)
```

```
X_test_data = tfidf.transform(X_test_data['text']).toarray()
```

```
y_test_pred = classifier.predict(X_test_data)
```

```
y_pred_labels = le.inverse_transform(y_test_pred)
y_pred_labels
```

```
array(['anticipation', 'anticipation', 'sadness', ..., 'anticipation',
       'joy', 'sadness'], dtype=object)
```

```
submission = pd.DataFrame({
        'id': test_data['tweet_id'],
        'emotion': y_pred_labels
})
```

```
submission.to_csv('/kaggle/working/submission.csv', index=False)
```

```
xx = pd.read_csv('/kaggle/working/submission.csv')
xx
```

```
              id       emotion
     0  0x28b412  anticipation
     1  0x2de201  anticipation
     2  0x218443       sadness
     3  0x2939d5           joy
     4  0x26289a         trust
   ...       ...           ...
411967  0x2913b4  anticipation
411968  0x2a980e  anticipation
411969  0x316b80  anticipation
411970  0x29d0cb           joy
411971  0x2a6a4f       sadness
411972 rows × 2 columns
```

Experience of other methods attemping

In the lab, I had also tried BERT, which is the best way to cope with text data I have known, to accomplish the competition. However, there were some problems I had met:

1. The memory on kaggle seemed to not enough to run, everytime I run on it the system was crash.
2. The training time was excessively long, even when using a smaller batch size or fewer epochs. This made it challenging to iterate and fine-tune the model efficiently.

Hence, I eventually choose the alternative way, Random Forest Classifier, to solve the problem, but I have thought some probable attemps to improve. For example, use lighter version of BERT, or simplify the procedure data processing. Though I haven't succeeded on this competition, I can so that better by this experience.

開始使用 AI 編寫或生成程式碼。