HOMEWORK 6: TEXT CLASSIFICATION

In this homework, you will create models to classify texts from TRUE call-center. There are two classification tasks:

1. Action Classification: Identify which action the customer would like to take (e.g. enquire, report, cancle)
2. Object Classification: Identify which object the customer is referring to (e.g. payment, truemoney, internet, roaming)

We will focus only on the Object Classification task for this homework.

In this homework, you are asked compare different text classification models in terms of accuracy and inference time.

You will need to build 3 different models.

1. A model based on tf-idf
2. A model based on MUSE
3. A model based on wangchanBERTa

You will be ask to submit 3 different files (.pdf from .ipynb) that does the 3 different models. Finally, answer the accuracy and runtime numbers in MCV.

This homework is quite free form, and your answer may vary. We hope that the processing during the course of this assignment will make you think more about the design choices in text classification.

```
!wget --no-check-certificate https://www.dropbox.com/s/37u83g55p19kvrl/clean-phone-data-for-students.csv -O ./clean-phone-
```

```
--2025-02-15 15:16:27--  https://www.dropbox.com/s/37u83g55p19kvrl/clean-phone-data-for-students.csv
Resolving www.dropbox.com (www.dropbox.com)... 2620:100:6035:18::a27d:5512, 162.125.85.18
Connecting to www.dropbox.com (www.dropbox.com)|2620:100:6035:18::a27d:5512|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://www.dropbox.com/scl/fi/8h8hvsw9uj6o0524lfe4i/clean-phone-data-for-students.csv?rlkey=lwv5xbf16jerehnv3
--2025-02-15 15:16:27--  https://www.dropbox.com/scl/fi/8h8hvsw9uj6o0524lfe4i/clean-phone-data-for-students.csv?rlkey=lw
Reusing existing connection to [www.dropbox.com]:443.
HTTP request sent, awaiting response... 302 Found
Location: https://ucce80429c1d534064b3a6d9f042.dl.dropboxusercontent.com/cd/0/inline/CkLrInUX8rJuXukZ2pmOzA1HWeZom4RNG8z
--2025-02-15 15:16:28--  https://ucce80429c1d534064b3a6d9f042.dl.dropboxusercontent.com/cd/0/inline/CkLrInUX8rJuXukZ2pmO
Resolving ucce80429c1d534064b3a6d9f042.dl.dropboxusercontent.com (ucce80429c1d534064b3a6d9f042.dl.dropboxusercontent.com
Connecting to ucce80429c1d534064b3a6d9f042.dl.dropboxusercontent.com (ucce80429c1d534064b3a6d9f042.dl.dropboxusercontent
HTTP request sent, awaiting response... 200 OK
Length: 2518977 (2.4M) [text/plain]
Saving to: ‘./clean-phone-data-for-students.csv’

./clean-phone-data- 100%[===================>]   2.40M  4.07MB/s    in 0.6s

2025-02-15 15:16:30 (4.07 MB/s) - ‘./clean-phone-data-for-students.csv’ saved [2518977/2518977]
```

```
!pip install -q pythainlp
```

Import Libs

```
%matplotlib inline
import pandas
import sklearn
import numpy as np
import time
import matplotlib.pyplot as plt
import pandas as pd
from pprint import pprint

from torch.utils.data import Dataset
from IPython.display import display
from collections import defaultdict
from sklearn.metrics import accuracy_score
```

Loading data

First, we load the data from disk into a Dataframe.

A Dataframe is essentially a table, or 2D-array/Matrix with a name for each column.

```
data_df = pd.read_csv('clean-phone-data-for-students.csv')
```

Let's preview the data.

```python
# Show the top 5 rows
display(data_df.head())
# Summarize the data
data_df.describe()
```

	Sentence Utterance	Action	Object
0	<PHONE_NUMBER_REMOVED> ผมไปจ่ายเงินที่ Counte...	enquire	payment
1	internet ยังความเร็วอยุ่เท่าไหร ครับ	enquire	package
2	ตะกี้ไปชำระค่าบริการไปแล้ว แต่ยังใช้งานไม่ได้...	report	suspend
3	พี่ค่ะยังใช้ internet ไม่ได้เลยค่ะ เป็นเครื่อ...	enquire	internet
4	ฮาโหล คะ พอดีว่าเมื่อวานเปิดซิมทรูมูฟ แต่มันโ...	report	phone_issues

	Sentence Utterance	Action	Object
count	16175	16175	16175
unique	13389	10	33
top	บริการอื่นๆ	enquire	service
freq	97	10377	2525

Data cleaning

We call the DataFrame.describe() again. Notice that there are 33 unique labels/classes for object and 10 unique labels for action that the model will try to predict. But there are unwanted duplications e.g. Idd,idd,lotalty_card,Lotalty_card

Also note that, there are 13389 unqiue sentence utterances from 16175 utterances. You have to clean that too!

#TODO 0.1:

You will have to remove unwanted label duplications as well as duplications in text inputs. Also, you will have to trim out unwanted whitespaces from the text inputs. This shouldn't be too hard, as you have already seen it in the demo.

```python
display(data_df.describe())
display(data_df.Object.unique())
display(data_df.Action.unique())
```

	Sentence Utterance	Action	Object
count	16175	16175	16175
unique	13389	10	33
top	บริการอื่นๆ	enquire	service
freq	97	10377	2525

```
array(['payment', 'package', 'suspend', 'internet', 'phone_issues',
       'service', 'nonTrueMove', 'balance', 'detail', 'bill', 'credit',
       'promotion', 'mobile_setting', 'iservice', 'roaming', 'truemoney',
       'information', 'lost_stolen', 'balance_minutes', 'idd',
       'TrueMoney', 'garbage', 'Payment', 'IDD', 'ringtone', 'Idd',
       'rate', 'loyalty_card', 'contact', 'officer', 'Balance', 'Service',
       'Loyalty_card'], dtype=object)
array(['enquire', 'report', 'cancel', 'Enquire', 'buy', 'activate',
       'request', 'Report', 'garbage', 'change'], dtype=object)
```

```python
clean_data_time = time.time()

# Group the duplicate label
data_df.dropna(subset=['Object'], inplace=True)
data_df['Object'] = data_df['Object'].apply(lambda x: x.lower())

# Clean the data
data_df['Sentence Utterance'] = data_df['Sentence Utterance'].apply(lambda x: str(x).strip())
data_df['Sentence Utterance'] = data_df['Sentence Utterance'].apply(lambda x: x.lower())
data_df.drop_duplicates(subset=['Sentence Utterance'], inplace=True)

# Drop the unused columns
data_df.drop(columns=['Action'], inplace=True)

clean_data_time = time.time() - clean_data_time
```

Split data into train, valdation, and test sets (normally the ratio will be 80:10:10 , respectively). We recommend to use train_test_spilt from scikit-learn to split the data into train, validation, test set.

In addition, it should split the data that distribution of the labels in train, validation, test set are similar. There is **stratify** option to handle this issue.

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

Make sure the same data splitting is used for all models.

```
from sklearn.model_selection import train_test_split
from collections import Counter

split_data_time = time.time()

# For the object column, we will only keep the object that has more than 2% of the total data
object_counter = Counter(data_df['Object'])
stratify_col = data_df['Object'].apply(lambda x: 'other' if object_counter[x] < 0.02*len(data_df) else x)
train_df, test_df = train_test_split(data_df, test_size=0.2, random_state=4242, stratify=stratify_col)

object_counter = Counter(test_df['Object'])
stratify_col = test_df['Object'].apply(lambda x: 'other' if object_counter[x] < 0.02*len(test_df) else x)
test_df, val_df = train_test_split(test_df, test_size=0.5, random_state=4242, stratify=stratify_col)

train_df = train_df.reset_index(drop=True)
test_df = test_df.reset_index(drop=True)
val_df = val_df.reset_index(drop=True)

print(f"Train size: {len(train_df)}")
print(f"Test size: {len(test_df)}")
print(f"Val size: {len(val_df)}")

split_data_time = time.time() - split_data_time
```

```
Train size: 10689
Test size: 1336
Val size: 1337
```

```
# Save the data
# train_df.to_csv('train.csv', index=False)
# test_df.to_csv('test.csv', index=False)
# val_df.to_csv('val.csv', index=False)
```

Model 2 MUSE

Build a simple logistic regression model using features from the MUSE model.

Which MUSE model will you use? Why?

Ans: `sentence-transformers/use-cmlm-multilingual` because it is easy to use and was trained on many languages(100+). The community also has a lot of good feedback on this model. (about 3,000 downloads last month)

MUSE is typically used with tensorflow. However, there are some pytorch conversions made by some people.

https://huggingface.co/sentence-transformers/use-cmlm-multilingual https://huggingface.co/dayyass/universal-sentence-encoder-multilingual-large-3-pytorch

```
!pip install -qU sentence-transformers
```

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score
from sentence_transformers import SentenceTransformer
```

```
train_df, test_df, val_df = pd.read_csv('train.csv'), pd.read_csv('test.csv'), pd.read_csv('val.csv')
```

```
encoder_model = SentenceTransformer('sentence-transformers/use-cmlm-multilingual')
```

```
Some weights of the model checkpoint at sentence-transformers/use-cmlm-multilingual were not used when initializing Bert
- This IS expected if you are initializing BertModel from the checkpoint of a model trained on another task or with anot
- This IS NOT expected if you are initializing BertModel from the checkpoint of a model that you expect to be exactly id
```

```
encoder_time = time.time()

train_x = encoder_model.encode(train_df['Sentence Utterance'].values, show_progress_bar=True)
test_x = encoder_model.encode(test_df['Sentence Utterance'].values, show_progress_bar=True)
val_x = encoder_model.encode(val_df['Sentence Utterance'].values, show_progress_bar=True)

train_y = train_df['Object']
```

```python
test_y = test_df['Object']
val_y = val_df['Object']

encoder_time = time.time() - encoder_time
```

```
Batches:   0%|          | 0/335 [00:00<?, ?it/s]
Batches:   0%|          | 0/42 [00:00<?, ?it/s]
Batches:   0%|          | 0/42 [00:00<?, ?it/s]
```

```python
train_time = time.time()

model = LogisticRegression(max_iter=1000, penalty='elasticnet', solver='saga', l1_ratio=0.5, C=1.0, n_jobs=-1)
model.fit(train_x, train_y)

train_time = time.time() - train_time
```

```python
all_inference_time = time.time()

# Predict the data
train_pred = model.predict(train_x)
test_pred = model.predict(test_x)
val_pred = model.predict(val_x)

all_inference_time = time.time() - all_inference_time

# Calculate the accuracy
train_acc = accuracy_score(train_y, train_pred)
test_acc = accuracy_score(test_y, test_pred)
val_acc = accuracy_score(val_y, val_pred)

print(f"Train accuracy: {train_acc}")
print(f"Test accuracy: {test_acc}")
print(f"Val accuracy: {val_acc}")
```

```
Train accuracy: 0.7254186546917392
Test accuracy: 0.7148203592814372
Val accuracy: 0.7120418848167539
```

```python
# Print the classification report
print("Val classification report")
print(classification_report(val_y, val_pred))

import pickle

# Save the classificaion report
classification_report_dict = classification_report(val_y, val_pred, output_dict=True)
with open('classification_report_muse.pkl', 'wb') as f:
    pickle.dump(classification_report_dict, f)
```

```
Val classification report
                 precision    recall  f1-score   support

        balance       0.73      0.83      0.78       148
balance_minutes       0.00      0.00      0.00         5
           bill       0.62      0.52      0.57        54
         credit       1.00      0.65      0.79        17
         detail       0.50      0.27      0.35        33
        garbage       0.00      0.00      0.00         6
            idd       0.91      0.71      0.80        14
    information       0.64      0.48      0.55        29
       internet       0.69      0.83      0.76       179
       iservice       0.00      0.00      0.00         3
    lost_stolen       0.96      0.77      0.85        30
   loyalty_card       1.00      0.25      0.40         8
 mobile_setting       0.57      0.14      0.23        28
    nontruemove       0.60      0.14      0.23        21
        officer       0.00      0.00      0.00         1
        package       0.65      0.75      0.70       179
        payment       0.69      0.72      0.70        64
   phone_issues       0.61      0.67      0.64        58
      promotion       0.72      0.70      0.71       115
           rate       0.00      0.00      0.00         6
       ringtone       0.88      0.64      0.74        11
        roaming       0.76      0.76      0.76        17
        service       0.77      0.84      0.80       211
        suspend       0.73      0.77      0.75        73
      truemoney       0.92      0.85      0.88        27

       accuracy                           0.71      1337
      macro avg       0.60      0.49      0.52      1337
   weighted avg       0.70      0.71      0.70      1337

/Users/jirayuwat/anaconda3/envs/nlp/lib/python3.11/site-packages/sklearn/metrics/_classification.py:1565: UndefinedMetri
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
/Users/jirayuwat/anaconda3/envs/nlp/lib/python3.11/site-packages/sklearn/metrics/_classification.py:1565: UndefinedMetri
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/jirayuwat/anaconda3/envs/nlp/lib/python3.11/site-packages/sklearn/metrics/_classification.py:1565: UndefinedMetri
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/jirayuwat/anaconda3/envs/nlp/lib/python3.11/site-packages/sklearn/metrics/_classification.py:1565: UndefinedMetri
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/jirayuwat/anaconda3/envs/nlp/lib/python3.11/site-packages/sklearn/metrics/_classification.py:1565: UndefinedMetri
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/jirayuwat/anaconda3/envs/nlp/lib/python3.11/site-packages/sklearn/metrics/_classification.py:1565: UndefinedMetri
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```python
print(f'''
All preprocessing time: {clean_data_time + split_data_time + encoder_time:.2f} seconds
 - Clean data time: {clean_data_time:.2f} seconds
 - Split data time: {split_data_time:.2f} seconds
 - Encoder time: {encoder_time:.2f} seconds
Training time: {train_time:.2f} seconds
Inference time: {all_inference_time:.2f} seconds
'''.strip())
```

```
All preprocessing time: 45.99 seconds
 - Clean data time: 0.01 seconds
 - Split data time: 0.01 seconds
 - Encoder time: 45.96 seconds
Training time: 24.70 seconds
Inference time: 0.01 seconds
```