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 13:57:38--  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 13:57:38--  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://uc7df6d5a23f54bd3592a4a478d5.dl.dropboxusercontent.com/cd/0/inline/CkIpCV84EQav89R1pHJhwTj_FwQ5tzBwdvR
--2025-02-15 13:57:39--  https://uc7df6d5a23f54bd3592a4a478d5.dl.dropboxusercontent.com/cd/0/inline/CkIpCV84EQav89R1pHJh
Resolving uc7df6d5a23f54bd3592a4a478d5.dl.dropboxusercontent.com (uc7df6d5a23f54bd3592a4a478d5.dl.dropboxusercontent.com
Connecting to uc7df6d5a23f54bd3592a4a478d5.dl.dropboxusercontent.com (uc7df6d5a23f54bd3592a4a478d5.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  4.04MB/s    in 0.6s

2025-02-15 13:57:41 (4.04 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.

```
# 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 ยังความเร็วอยู่เท่าไหร 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.

```
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)
```

```
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 1 TF-IDF

Build a model to train a tf-idf text classifier. Use a simple logistic regression model for the classifier.

For this part, you may find this tutorial helpful.

Below are some design choices you need to consider to accomplish this task. Be sure to answer them when you submit your model.

What tokenizer will you use? Why?

Ans: Use `newmm` tokenizer from PyThaiNLP. It is a dictionary-based tokenizer that is suitable for Thai text. Additionally, I use `ngram_range=(1, 2)` to include bigram in the feature set. This is because bigram can capture the relationship between words that are close to each other.

Will you ignore some stop words (a, an, the, to, etc. for English) in your tf-idf? Is it important? PythaiNLP provides a list of stopwords if you want to use (https://pythainlp.org/docs/2.0/api/corpus.html#pythainlp.corpus.common.thai_stopwords)

Ans: I use the list of stopwords provided by PyThaiNLP. It is important to remove stopwords because they are common words that do not carry much meaning. Removing them can help the model focus on more important words.

The dictionary of TF-IDF is usually based on the training data. How many words in the test set are OOVs?

Ans: 324 words

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

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score
import pythainlp
```

```
tokenize_and_vectorize_time = time.time()

# Create the tf-idf vectorizer
tf_idf = TfidfVectorizer(tokenizer=pythainlp.word_tokenize,
                         ngram_range=(1, 2),
                         stop_words=list(pythainlp.corpus.thai_stopwords()))
tf_idf.fit(train_df['Sentence Utterance'])
```

```

# Transform the data
train_x = tf_idf.transform(train_df['Sentence Utterance'])
test_x = tf_idf.transform(test_df['Sentence Utterance'])
val_x = tf_idf.transform(val_df['Sentence Utterance'])

tokenize_and_vectorize_time = time.time() - tokenize_and_vectorize_time

# Get y label
train_y = train_df['Object']
test_y = test_df['Object']
val_y = val_df['Object']
```

```
/Users/jirayuwat/anaconda3/envs/nlp/lib/python3.11/site-packages/sklearn/feature_extraction/text.py:517: UserWarning: Th
  warnings.warn(
/Users/jirayuwat/anaconda3/envs/nlp/lib/python3.11/site-packages/sklearn/feature_extraction/text.py:402: UserWarning: Yo
  warnings.warn(
```

```
train_time = time.time()

# Create the model
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
```

```
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: 0.7176536626438395
Test accuracy: 0.6803892215568862
Val accuracy: 0.6798803290949887
```

```
# 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_tfidf.pkl', 'wb') as f:
    pickle.dump(classification_report_dict, f)
```

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

        balance       0.75      0.78      0.76       148
balance_minutes       0.80      0.80      0.80         5
           bill       0.65      0.48      0.55        54
         credit       1.00      0.76      0.87        17
         detail       0.89      0.24      0.38        33
        garbage       0.00      0.00      0.00         6
            idd       0.82      0.64      0.72        14
    information       1.00      0.31      0.47        29
       internet       0.66      0.82      0.73       179
       iservice       0.00      0.00      0.00         3
    lost_stolen       0.93      0.90      0.92        30
   loyalty_card       1.00      0.62      0.77         8
 mobile_setting       0.85      0.39      0.54        28
    nontruemove       0.89      0.38      0.53        21
        officer       0.00      0.00      0.00         1
        package       0.62      0.72      0.67       179
        payment       0.58      0.66      0.61        64
   phone_issues       0.63      0.57      0.60        58
      promotion       0.56      0.55      0.56       115
           rate       0.00      0.00      0.00         6
       ringtone       1.00      0.91      0.95        11
```

```
     roaming       0.85      0.65      0.73        17
     service       0.66      0.82      0.73       211
     suspend       0.70      0.67      0.69        73
  truemoney       0.83      0.74      0.78        27

    accuracy                           0.68      1337
   macro avg       0.67      0.54      0.57      1337
weighted avg       0.69      0.68      0.67      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 + tokenize_and_vectorize_time:.2f} seconds
 - Clean data time: {clean_data_time:.2f} seconds
 - Split data time: {split_data_time:.2f} seconds
 - Tokenize and vectorize time: {tokenize_and_vectorize_time:.2f} seconds
Train time: {train_time:.2f} seconds
Inference time: {all_inference_time:.2f} seconds
'''.strip())
```

```
All preprocessing time: 1.20 seconds
 - Clean data time: 0.02 seconds
 - Split data time: 0.03 seconds
 - Tokenize and vectorize time: 1.15 seconds
Train time: 34.22 seconds
Inference time: 0.01 seconds
```

```python
# Count OOV words
tokenizer = pythainlp.word_tokenize

train_words = set()
for sentence in train_df['Sentence Utterance']:
    train_words.update(tokenizer(sentence))

test_words = set()
for sentence in test_df['Sentence Utterance']:
    test_words.update(tokenizer(sentence))

val_words = set()
for sentence in val_df['Sentence Utterance']:
    val_words.update(tokenizer(sentence))

oov_words = test_words.union(val_words) - train_words
print(f"OOV words: {len(oov_words)}")
print("OOV words:", oov_words)
```

```
OOV words: 324
OOV words: {'เหล่านี้', 'ไทร', 'เหตุการณ์', 'เชียงคาน', 'สเวิด', 'ประวัติการ', 'การผ่อน', 'lift', 'คล่องตัว', '4541843', 'ละคร', 'เล
```