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 09:30:32--  https://www.dropbox.com/s/37u83g55p19kvrl/clean-phone-data-for-students.csv
Resolving www.dropbox.com (www.dropbox.com)... 162.125.2.18, 2620:100:6017:18::a27d:212
Connecting to www.dropbox.com (www.dropbox.com)|162.125.2.18|: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 09:30:32--  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://ucedf0b5923cd38339989bc62a9f.dl.dropboxusercontent.com/cd/0/inline/CkJvDe1QdFc6hKiyZjSdzab3ahqgm-dltXL
--2025-02-15 09:30:32--  https://ucedf0b5923cd38339989bc62a9f.dl.dropboxusercontent.com/cd/0/inline/CkJvDe1QdFc6hKiyZjSd
Resolving ucedf0b5923cd38339989bc62a9f.dl.dropboxusercontent.com (ucedf0b5923cd38339989bc62a9f.dl.dropboxusercontent.com
Connecting to ucedf0b5923cd38339989bc62a9f.dl.dropboxusercontent.com (ucedf0b5923cd38339989bc62a9f.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  --.-KB/s    in 0.05s

2025-02-15 09:30:33 (46.7 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 3 WangchanBERTa

We ask you to train a WangchanBERTa-based model.

We recommend you use the thaixtransformers fork (which we used in the PoS homework). https://github.com/PyThaiNLP/thaixtransformers

The structure of the code will be very similar to the PoS homework. You will also find the huggingface tutorial useful. Or you can also add a softmax layer by yourself just like in the previous homework.

Which WangchanBERTa model will you use? Why? (Don't forget to clean your text accordingly).

Ans:

```
%pip install -q wandb
%pip install -q transformers==4.30.1 datasets evaluate thaixtransformers
%pip install -q emoji pythainlp sefr_cut tinydb seqeval sentencepiece pydantic jsonlines
%pip install -q peft==0.10.0
```

```
import pandas as pd
from thaixtransformers import Tokenizer
from transformers import AutoModelForSequenceClassification, DataCollatorWithPadding, TrainingArguments, Trainer
from datasets import Dataset
import time
from sklearn.metrics import accuracy_score
```

```
create_dataset_time = time.time()

train_df, test_df, val_df = pd.read_csv('train.csv'), pd.read_csv('test.csv'), pd.read_csv('val.csv')
train_df.columns = ['text', 'label']
test_df.columns = ['text', 'label']
val_df.columns = ['text', 'label']

label2id = {label: i for i, label in enumerate(sorted(train_df['label'].unique()))}
id2label = {i: label for label, i in label2id.items()}
train_df['label'] = train_df['label'].apply(lambda x: label2id[x])
test_df['label'] = test_df['label'].apply(lambda x: label2id[x])
val_df['label'] = val_df['label'].apply(lambda x: label2id[x])

```

```
# Create dataset
train_dataset = Dataset.from_pandas(train_df)
test_dataset = Dataset.from_pandas(test_df)
val_dataset = Dataset.from_pandas(val_df)
```

```
tokenizer = Tokenizer("airesearch/wangchanberta-base-wiki-newmm")
model = AutoModelForSequenceClassification.from_pretrained("airesearch/wangchanberta-base-wiki-newmm",
                                                           num_labels=train_df['label'].max()+1,
                                                           label2id=label2id,
                                                           id2label=id2label)

data_collator = DataCollatorWithPadding(tokenizer=tokenizer)
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/file_download.py:795: FutureWarning: `resume_download` is deprec
  warnings.warn(
The tokenizer class you load from this checkpoint is not the same type as the class this function is called from. It may
The tokenizer class you load from this checkpoint is 'RobertaTokenizer'.
The class this function is called from is 'ThaiWordsNewmmTokenizer'.
/usr/local/lib/python3.11/dist-packages/transformers/modeling_utils.py:463: FutureWarning: You are using `torch.load` wi
  return torch.load(checkpoint_file, map_location="cpu")
Some weights of the model checkpoint at airesearch/wangchanberta-base-wiki-newmm were not used when initializing Roberta
- This IS expected if you are initializing RobertaForSequenceClassification from the checkpoint of a model trained on an
- This IS NOT expected if you are initializing RobertaForSequenceClassification from the checkpoint of a model that you
Some weights of RobertaForSequenceClassification were not initialized from the model checkpoint at airesearch/wangchanbe
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

```
# Tokenize the data
train_dataset = train_dataset.map(lambda x: tokenizer(x['text'], padding="max_length", truncation=True), batched=True)
test_dataset = test_dataset.map(lambda x: tokenizer(x['text'], padding="max_length", truncation=True), batched=True)
val_dataset = val_dataset.map(lambda x: tokenizer(x['text'], padding="max_length", truncation=True), batched=True)

create_dataset_time = time.time() - create_dataset_time
```

```
Map: 100%                                   10689/10689 [00:09<00:00, 1229.44 examples/s]
Asking to pad to max_length but no maximum length is provided and the model has no predefined maximum length. Default to
Asking to truncate to max_length but no maximum length is provided and the model has no predefined maximum length. Defau
Map: 100%                                   1336/1336 [00:01<00:00, 875.73 examples/s]
Map: 100%                                   1337/1337 [00:01<00:00, 1036.96 examples/s]
```

```
def compute_metrics(pred):
    labels = pred.label_ids
    preds = pred.predictions.argmax(-1)
    return {"accuracy": accuracy_score(labels, preds)}

training_args = TrainingArguments(
    #########################
    output_dir="text_classification",
    learning_rate=2e-5,
    num_train_epochs=2,
    weight_decay=0.01,
    push_to_hub=False
    #########################
)

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

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

```
/usr/local/lib/python3.11/dist-packages/transformers/optimization.py:411: FutureWarning: This implementation of AdamW is
  warnings.warn(
```

[2674/2674 08:25, Epoch 2/2]

Step	Training Loss
500	1.898900
1000	1.184500
1500	1.021700
2000	0.759500
2500	0.766700

```python
all_inference_time = time.time()

# Predict the data
train_pred = trainer.predict(train_dataset)
test_pred = trainer.predict(test_dataset)
val_pred = trainer.predict(val_dataset)

all_inference_time = time.time() - all_inference_time

# Calculate the accuracy
train_acc = train_pred.metrics['test_accuracy']
test_acc = test_pred.metrics['test_accuracy']
val_acc = val_pred.metrics['test_accuracy']

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

```
Train accuracy: 0.8529329216952006
Test accuracy: 0.7574850299401198
Val accuracy: 0.7471952131637996
```

```python
from sklearn.metrics import classification_report, accuracy_score

# Print the classification report
print("Val classification report")
print(classification_report(val_pred.label_ids, val_pred.predictions.argmax(-1)))

import pickle

# Save the classificaion report
classification_report_dict = classification_report(val_pred.label_ids, val_pred.predictions.argmax(-1), output_dict=True)
with open('classification_report_wangchanberta.pkl', 'wb') as f:
    pickle.dump(classification_report_dict, f)
```

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

           0       0.82      0.80      0.81       148
           1       0.00      0.00      0.00         5
           2       0.61      0.63      0.62        54
           4       1.00      0.88      0.94        17
           5       0.57      0.24      0.34        33
           6       0.00      0.00      0.00         6
           7       0.75      0.64      0.69        14
           8       0.62      0.55      0.58        29
           9       0.76      0.79      0.77       179
          10       0.00      0.00      0.00         3
          11       1.00      0.93      0.97        30
          12       1.00      0.38      0.55         8
          13       0.46      0.43      0.44        28
          14       1.00      0.14      0.25        21
          15       0.00      0.00      0.00         1
          16       0.71      0.82      0.76       179
          17       0.65      0.75      0.70        64
          18       0.69      0.72      0.71        58
          19       0.74      0.70      0.72       115
          20       0.00      0.00      0.00         6
          21       0.92      1.00      0.96        11
          22       0.67      0.71      0.69        17
          23       0.79      0.86      0.82       211
          24       0.78      0.86      0.82        73
          25       0.90      0.96      0.93        27

    accuracy                           0.75      1337
   macro avg       0.62      0.55      0.56      1337
weighted avg       0.74      0.75      0.73      1337
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

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

```
All preprocessing time: 20.37 seconds
 - Clean data time: 0.03 seconds
 - Split data time: 0.05 seconds
 - Create dataset time: 20.29 seconds
Training time: 507.56 seconds
Inference time: 32.75 seconds
```

After you

Comparison

After you have completed the 3 models, compare the accuracy, ease of implementation, and inference speed (from cleaning, tokenization, till model compute) between the three models in mycourseville.