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.

In [16]:
```
# !wget --no-check-certificate https://www.dropbox.com/s/37u83g55p19kvrl/clean-p
```

In [17]:
```
# !pip install pythainlp
```

Import Libs

In [18]:
```
# %matplotlib inline
# import pandas
# import sklearn
# import numpy as np
# import matplotlib.pyplot as plt
# import pandas as pd

# from torch.utils.data import Dataset
# from IPython.display import display
# from collections import defaultdict
# from sklearn.metrics import accuracy_score
# from sklearn.model_selection import train_test_split
# from pythainlp.tokenize import word_tokenize
# from sklearn.feature_extraction.text import TfidfVectorizer
# from sklearn.linear_model import LogisticRegression
# from sklearn.pipeline import Pipeline
# from pythainlp.corpus.common import thai_stopwords
# import time
# import torch
# from sentence_transformers import SentenceTransformer
# from sklearn.linear_model import LogisticRegression
# from sklearn.pipeline import make_pipeline
# from sklearn.preprocessing import StandardScaler
```

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.

In [19]:
```
data_df = pd.read_csv('clean-phone-data-for-students.csv')
```

Let's preview the data.

In [20]:
```
# 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

Out[20]:

	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.

```
In [21]: 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)
```

```
In [22]: data_df.columns
```

```
Out[22]: Index(['Sentence Utterance', 'Action', 'Object'], dtype='object')
```

```
In [23]: cols = ["Sentence Utterance", "Object"]
         data_df = data_df[cols]
         data_df.columns = ["input", "raw_label"]

         data_df["clean_label"]=data_df["raw_label"].str.lower().copy()
         data_df.drop("raw_label", axis=1, inplace=True)

         data_df["input"] = data_df["input"].str.strip()

         data_df = data_df.drop_duplicates(subset=['input'], keep='first')
```

```
In [24]: display(data_df["clean_label"].unique())
         display(data_df.describe())
         display(data_df.head())
```

```
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', 'garbage',
       'ringtone', 'rate', 'loyalty_card', 'contact', 'officer'],
      dtype=object)
```

	input	clean_label
count	13367	13367
unique	13367	26
top	สอบถามโปรโมชั่นปัจจุบันที่ใช้อยู่ค่ะ	service
freq	1	2108

	input	clean_label
0	<PHONE_NUMBER_REMOVED> ผมไปจ่ายเงินที่ Counter...	payment
1	internet ยังความเร็วอยุ่เท่าไหร ครับ	package
2	ตะกี้ไปชำระค่าบริการไปแล้ว แต่ยังใช้งานไม่ได้ ค่ะ	suspend
3	พี่ค่ะยังใช้ internet ไม่ได้เลยค่ะ เป็นเครื่อง...	internet
4	ฮาโหล คะ พอดีว่าเมื่อวานเปิดซิมทรูมูฟ แต่มันโท...	phone_issues

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.

In [25]:
```
data_x = np.array(list(data_df["input"]))
data_y_tmp = np.array(list(data_df["clean_label"]))
data_y = []

map_label_num = {y.strip():i for i,y in enumerate(list(data_df["clean_label"].ur
map_num_label = {i:y.strip() for i,y in enumerate(list(data_df["clean_label"].ur

for i in range(len(data_y_tmp)):
    data_y.append(int(map_label_num[data_y_tmp[i]]))
data_y = np.array(data_y)
print(len(data_y))
```

```
13367
```

In [26]:
```
unique, counts = np.unique(data_y, return_counts=True)
valid_classes = unique[counts >= 10]
valid_indices = np.isin(data_y, valid_classes)
data_x,data_y  = data_x[valid_indices],data_y[valid_indices]
```

In [27]:
```
X_train, X_temp, y_train, y_temp = train_test_split(data_x, data_y, test_size=0.
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.50,

print("Train size:", len(X_train))
print("Validation size:", len(X_val))
print("Test size:",len(X_test))
```

```
Train size: 10690
Validation size: 1336
Test size: 1337
```

Model 2 MUSE

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

Which MUSE model will you use? Why?

Ans:

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

In [32]:
```
muse_model = SentenceTransformer("sentence-transformers/use-cmlm-multilingual")

def encode_text(sentences):
    return muse_model.encode(sentences, convert_to_numpy=True)

X_train_emb = encode_text(X_train)
X_val_emb = encode_text(X_val)
X_test_emb = encode_text(X_test)

log_reg = LogisticRegression(random_state=69)
start =time.time()
log_reg.fit(X_train_emb, y_train)
end =time.time()

train_acc = log_reg.score(X_train_emb, y_train)
val_acc = log_reg.score(X_val_emb, y_val)
test_acc = log_reg.score(X_test_emb, y_test)

print(f"Training Time: {end - start:.4f} seconds")
print(f"Train Accuracy: {train_acc:.4f}")
print(f"Validation Accuracy: {val_acc:.4f}")
print(f"Test Accuracy: {test_acc:.4f}")
```

```
Some weights of the model checkpoint at sentence-transformers/use-cmlm-multilingu
al were not used when initializing BertModel: ['cls.predictions.bias', 'cls.predi
ctions.transform.LayerNorm.bias', 'cls.predictions.transform.LayerNorm.weight',
'cls.predictions.transform.dense.bias', 'cls.predictions.transform.dense.weight',
'cls.seq_relationship.bias', 'cls.seq_relationship.weight']
- This IS expected if you are initializing BertModel from the checkpoint of a mod
el trained on another task or with another architecture (e.g. initializing a Bert
ForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing BertModel from the checkpoint of a
model that you expect to be exactly identical (initializing a BertForSequenceClas
sification model from a BertForSequenceClassification model).
```

```
Batches:   0%|          | 0/335 [00:00<?, ?it/s]
Batches:   0%|          | 0/42 [00:00<?, ?it/s]
Batches:   0%|          | 0/42 [00:00<?, ?it/s]
Training Time: 2.2055 seconds
Train Accuracy: 0.7351
Validation Accuracy: 0.7118
Test Accuracy: 0.7023
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: Co
nvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```