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 [46]:

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

In [47]:

```
# !pip install pythainlp
```

Import Libs

In [48]:

```
# %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 transformers import AutoTokenizer, AutoModelForSequenceClassification, Tr
# from datasets import Dataset
# from sklearn.preprocessing import LabelEncoder
```

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

Let's preview the data.

```
In [50]: # 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[50]:

	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 [51]:

```
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 [52]:

```
data_df.columns
```

Out[52]:

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

In [53]:

```
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 [54]:

```
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 [55]:
```
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"].un
map_num_label = {i:y.strip() for i,y in enumerate(list(data_df["clean_label"].un

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 [56]:
```
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 [57]:
```
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 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:

In [58]:
```
!pip install wandb
```

```
Requirement already satisfied: wandb in /usr/local/lib/python3.10/dist-packages
(0.19.1)
Requirement already satisfied: click!=8.0.0,>=7.1 in /usr/local/lib/python3.10/di
st-packages (from wandb) (8.1.7)
Requirement already satisfied: docker-pycreds>=0.4.0 in /usr/local/lib/python3.1
0/dist-packages (from wandb) (0.4.0)
Requirement already satisfied: gitpython!=3.1.29,>=1.0.0 in /usr/local/lib/python
3.10/dist-packages (from wandb) (3.1.43)
Requirement already satisfied: platformdirs in /usr/local/lib/python3.10/dist-pac
kages (from wandb) (4.3.6)
Requirement already satisfied: protobuf!=4.21.0,!=5.28.0,<6,>=3.19.0 in /usr/loca
l/lib/python3.10/dist-packages (from wandb) (3.20.3)
Requirement already satisfied: psutil>=5.0.0 in /usr/local/lib/python3.10/dist-pa
ckages (from wandb) (5.9.5)
Requirement already satisfied: pydantic<3,>=2.6 in /usr/local/lib/python3.10/dist
-packages (from wandb) (2.11.0a1)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.10/dist-packages
(from wandb) (6.0.2)
Requirement already satisfied: requests<3,>=2.0.0 in /usr/local/lib/python3.10/di
st-packages (from wandb) (2.32.3)
Requirement already satisfied: sentry-sdk>=2.0.0 in /usr/local/lib/python3.10/dis
t-packages (from wandb) (2.19.2)
Requirement already satisfied: setproctitle in /usr/local/lib/python3.10/dist-pac
kages (from wandb) (1.3.4)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packa
ges (from wandb) (75.1.0)
Requirement already satisfied: typing-extensions<5,>=4.4 in /usr/local/lib/python
3.10/dist-packages (from wandb) (4.12.2)
Requirement already satisfied: six>=1.4.0 in /usr/local/lib/python3.10/dist-packa
ges (from docker-pycreds>=0.4.0->wandb) (1.17.0)
Requirement already satisfied: gitdb<5,>=4.0.1 in /usr/local/lib/python3.10/dist-
packages (from gitpython!=3.1.29,>=1.0.0->wandb) (4.0.11)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.1
0/dist-packages (from pydantic<3,>=2.6->wandb) (0.7.0)
Requirement already satisfied: pydantic-core==2.28.0 in /usr/local/lib/python3.1
0/dist-packages (from pydantic<3,>=2.6->wandb) (2.28.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python
3.10/dist-packages (from requests<3,>=2.0.0->wandb) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-pac
kages (from requests<3,>=2.0.0->wandb) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/di
st-packages (from requests<3,>=2.0.0->wandb) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/di
st-packages (from requests<3,>=2.0.0->wandb) (2025.1.31)
Requirement already satisfied: smmap<6,>=3.0.1 in /usr/local/lib/python3.10/dist-
packages (from gitdb<5,>=4.0.1->gitpython!=3.1.29,>=1.0.0->wandb) (5.0.1)
```

In [59]:

```
from kaggle_secrets import UserSecretsClient
import wandb
user_secrets = UserSecretsClient()

my_secret = user_secrets.get_secret("wandb_api_key")

wandb.login(key=my_secret)
```

```
wandb: WARNING Calling wandb.login() after wandb.init() has no effect.
```

Out[59]:

```
True
```

In [60]:

```
import torch
import pandas as pd
from transformers import AutoTokenizer, AutoModelForSequenceClassification, Trai
from datasets import Dataset
from sklearn.preprocessing import LabelEncoder

# Load tokenizer and model
model_name = "airesearch/wangchanberta-base-att-spm-uncased"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSequenceClassification.from_pretrained(model_name, num_label

# Encode labels
label_encoder = LabelEncoder()
y_train_enc = label_encoder.fit_transform(y_train)
y_val_enc = label_encoder.transform(y_val)
y_test_enc = label_encoder.transform(y_test)

# Tokenize data
def tokenize_function(examples):
    return tokenizer(examples["text"], padding="max_length", truncation=True, ma

train_data = Dataset.from_dict({"text": X_train, "label": y_train_enc}).map(toke
val_data = Dataset.from_dict({"text": X_val, "label": y_val_enc}).map(tokenize_f
test_data = Dataset.from_dict({"text": X_test, "label": y_test_enc}).map(tokeniz

# Define training arguments
training_args = TrainingArguments(
    output_dir="./results",   # Keep output directory for saving checkpoints
    run_name="wangchanberta_classification",  # Set a different name for W&B
    eval_strategy="epoch",
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=3,
    save_strategy="epoch",
    save_total_limit=1,
    logging_dir="./logs",
    logging_steps=50,
    load_best_model_at_end=True
)

# Trainer

def compute_metrics(eval_pred):
    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=-1)  # Get the predicted class
    acc = accuracy_score(labels, predictions)  # Compute accuracy
    return {"accuracy": acc}

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_data,
    eval_dataset=val_data,
    compute_metrics=compute_metrics  # Add the metrics function here
)


# Train model
start = time.time()
```

```
trainer.train()
end = time.time()
```

```
Some weights of CamembertForSequenceClassification were not initialized from the
model checkpoint at airesearch/wangchanberta-base-att-spm-uncased and are newly i
nitialized: ['classifier.dense.bias', 'classifier.dense.weight', 'classifier.out_
proj.bias', 'classifier.out_proj.weight']
You should probably TRAIN this model on a down-stream task to be able to use it f
or predictions and inference.
Map:   0%|          | 0/10690 [00:00<?, ? examples/s]
Map:   0%|          | 0/1336 [00:00<?, ? examples/s]
Map:   0%|          | 0/1337 [00:00<?, ? examples/s]
```

[2007/2007 07:05, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Accuracy
1	1.338700	1.297383	0.618263
2	1.149300	1.343262	0.589072
3	0.610100	0.750168	0.773204

In [66]:

```
train_results = trainer.evaluate(train_data)
val_results = trainer.evaluate(val_data)
test_results = trainer.evaluate(test_data)
```

In [67]:

```
print(f"Training Time: {end - start:.4f} seconds")
print(f"Train Accuracy: {train_results['eval_accuracy']:.4f}")
print(f"Validation Accuracy: {val_results['eval_accuracy']:.4f}")
print(f"Test Accuracy: {test_results['eval_accuracy']:.4f}")
```

```
Training Time: 426.8042 seconds
Train Accuracy: 0.8536
Validation Accuracy: 0.7732
Test Accuracy: 0.7644
```

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.

Model1

- Training time: 3.0621 seconds
- Train Accuracy: 0.7650
- Validation Accuracy: 0.6939
- Test Accuracy: 0.6971

Model 2

- Training Time: 2.2055 seconds
- Train Accuracy: 0.7351
- Validation Accuracy: 0.7118

- Test Accuracy: 0.7023

Model 3

- Training Time: 426.8042 seconds
- Train Accuracy: 0.8536
- Validation Accuracy: 0.7732
- Test Accuracy: 0.7644

ANS

WangchanBERTa ดีที่สุด เพราะ มีaccuracyสูงสุดและเรากำลังทำCallCenterChatbotซึ่งไม่จำเป็นต้องเร็วมากนั้น