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 [18]:
```
!wget --no-check-certificate https://www.dropbox.com/s/37u83g55p19kvrl/clean-pho
```

```
--2025-02-14 14:20:18--  https://www.dropbox.com/s/37u83g55p19kvrl/clean-phone-da
ta-for-students.csv
Resolving www.dropbox.com (www.dropbox.com)... 162.125.66.18, 2620:100:6022:18::a
27d:4212
Connecting to www.dropbox.com (www.dropbox.com)|162.125.66.18|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://www.dropbox.com/scl/fi/8h8hvsw9uj6o0524lfe4i/clean-phone-data-f
or-students.csv?rlkey=lwv5xbf16jerehnv3lfgq5ue6 [following]
--2025-02-14 14:20:18--  https://www.dropbox.com/scl/fi/8h8hvsw9uj6o0524lfe4i/cle
an-phone-data-for-students.csv?rlkey=lwv5xbf16jerehnv3lfgq5ue6
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uccc63f7805c29a229279cb3a957.dl.dropboxusercontent.com/cd/0/inl
ine/CkFihJzRlWn7Gfe-dFq-QEKjqULHVvdtego467dU0CcDolaVwlGXo74DoUKfqqGSU-7rawx5pLyHw
F3GEByDVsziO0rMzE26KDaIuk_w1OvaHc_4-dg9apaDNG9OBvrE6Io/file# [following]
--2025-02-14 14:20:19--  https://uccc63f7805c29a229279cb3a957.dl.dropboxuserconte
nt.com/cd/0/inline/CkFihJzRlWn7Gfe-dFq-QEKjqULHVvdtego467dU0CcDolaVwlGXo74DoUKfqq
GSU-7rawx5pLyHwF3GEByDVsziO0rMzE26KDaIuk_w1OvaHc_4-dg9apaDNG9OBvrE6Io/file
Resolving uccc63f7805c29a229279cb3a957.dl.dropboxusercontent.com (uccc63f7805c29a
229279cb3a957.dl.dropboxusercontent.com)... 162.125.66.15, 2620:100:6022:15::a27
d:420f
Connecting to uccc63f7805c29a229279cb3a957.dl.dropboxusercontent.com (uccc63f7805
c29a229279cb3a957.dl.dropboxusercontent.com)|162.125.66.15|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2518977 (2.4M) [text/plain]
Saving to: ‘clean-phone-data-for-students.csv.1’

clean-phone-data-fo 100%[===================>]   2.40M  --.-KB/s    in 0.07s

2025-02-14 14:20:19 (34.8 MB/s) - ‘clean-phone-data-for-students.csv.1’ saved [25
18977/2518977]
```

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

Import Libs

In [ ]:
```
%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
```

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 [21]:
```
data_df = pd.read_csv('clean-phone-data-for-students.csv')
```

Let's preview the data.

In [22]:
```
# 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

Out[22]:

	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 [23]:
```
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 [24]:
```
data_df.columns
```

Out[24]:
```
Index(['Sentence Utterance', 'Action', 'Object'], dtype='object')
```

In [25]:
```
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 [26]:
```
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 ยังความเร็วอยู่เท่าไหร 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 [27]:
```
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 [28]:
```
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 [29]:
```
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 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:

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:

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

Ans:

In [30]:
```
def thai_tokenizer(text):
    return word_tokenize(text, keep_whitespace=False)
```

In [31]:
```
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 numpy as np
import time

# Define stopwords
stopwords = list(thai_stopwords())

# Initialize TF-IDF Vectorizer
vectorizer = TfidfVectorizer(
    tokenizer=None,  # Use default tokenization
    stop_words=stopwords,  # Remove Thai stopwords
    max_features=10000  # Limit vocabulary size to avoid overfitting
)

# Create a logistic regression pipeline
model = Pipeline([
    ("tfidf", vectorizer),
    ("classifier", LogisticRegression(max_iter=1000, random_state=42))
])

# Train model
start = time.time()
model.fit(X_train, y_train)
end = time.time()
# Evaluate model
print(f"Training time: {end - start:.4f} seconds")
train_acc = model.score(X_train, y_train)
```

```
val_acc = model.score(X_val, y_val)
test_acc = model.score(X_test, y_test)

print(f"Train Accuracy: {train_acc:.4f}")
print(f"Validation Accuracy: {val_acc:.4f}")
print(f"Test Accuracy: {test_acc:.4f}")

# Check OOV words (words in test set not seen in train set)
train_vocab = set(vectorizer.get_feature_names_out())
test_vocab = set(word for sentence in X_test for word in sentence.split())
oov_words = test_vocab - train_vocab
print(f"Number of OOV words: {len(oov_words)}")
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:409: U
serWarning: Your stop_words may be inconsistent with your preprocessing. Tokenizi
ng the stop words generated tokens ['กคน', 'กคร', 'กครา', 'กครา', 'กจะ', 'กช',
'กต', 'กท', 'กทาง', 'กน', 'กระท', 'กระน', 'กระไร', 'กล', 'กว', 'กส', 'กหน', 'ก
อ', 'กอย', 'กำล', 'กเม', 'กแห', 'กๆ', 'ขณะท', 'ขณะน', 'ขณะหน', 'ขณะเด', 'คงอ
ย', 'คร', 'ครบคร', 'ครบถ', 'คราท', 'คราน', 'คราวก', 'คราวท', 'คราวน', 'คราวหน',
'คราวหล', 'คราวโน', 'คราหน', 'คล', 'งก', 'งกระน', 'งกล', 'งกว', 'งข', 'งคง', 'งค
น', 'งคร', 'งคราว', 'งง', 'งจ', 'งจน', 'งจะ', 'งจาก', 'งต', 'งท', 'งน', 'งบ', 'งป
วง', 'งมวล', 'งละ', 'งว', 'งส', 'งหน', 'งหมด', 'งหมาย', 'งหล', 'งหลาย', 'งอย',
'งเก', 'งเคย', 'งเน', 'งเป', 'งเม', 'งแก', 'งแต', 'งแม', 'งแล', 'งโง', 'งโน', 'งใ
ด', 'งใหญ', 'งไง', 'งได', 'งไหน', 'งๆ', 'งๆจ', 'จก', 'จจ', 'จนกระท', 'จนกว', 'จน
ขณะน', 'จนถ', 'จนท', 'จนบ', 'จนเม', 'จนแม', 'จร', 'จรดก', 'จวนเจ', 'จวบก', 'จส',
'จสมบ', 'จะได', 'จากน', 'จำเป', 'จแล', 'ฉะน', 'ชะก', 'ชะจนกระท', 'ชะจนถ', 'ณๆ',
'ดการ', 'ดงาน', 'ดดล', 'ดต', 'ดทำ', 'ดน', 'ดว', 'ดหน', 'ดหา', 'ดเด', 'ดเผย', 'ด
แจง', 'ดให', 'ดไป', 'ดๆ', 'ตลอดถ', 'ตลอดท', 'ตลอดป', 'ตลอดว', 'ตามด', 'ตามท',
'ตามแต', 'ทว', 'ทำให', 'นก', 'นการ', 'นกาลนาน', 'นควร', 'นจะ', 'นด', 'นต', 'น
ท', 'นน', 'นนะ', 'นนาน', 'นมา', 'นมาก', 'นย', 'นยง', 'นยาว', 'นละ', 'นว', 'นวา
น', 'นอ', 'นอกจากท', 'นอกจากน', 'นอกจากว', 'นอกน', 'นอกเหน', 'นอาท', 'นา', 'นเคย',
'นเด', 'นเถอะ', 'นเน', 'นเป', 'นเพ', 'นเพราะ', 'นเพราะว', 'นเม', 'นเอง', 'นแก',
'นแต', 'นและก', 'นแหละ', 'นใด', 'นใดน', 'นไง', 'นได', 'นไป', 'นไร', 'นไว', 'นไห
น', 'นไหม', 'นๆ', 'บจากน', 'บต', 'บรอง', 'บว', 'บอกว', 'บอกแล', 'บางกว', 'บางค
ร', 'บางท', 'บางแห', 'บเน', 'บแต', 'ปฏ', 'ปร', 'ประการฉะน', 'ประการหน', 'ปรากฏว',
'พบว', 'พร', 'พวกก', 'พวกค', 'พวกฉ', 'พวกท', 'พวกน', 'พวกม', 'พวกโน', 'พอก',
'พอด', 'พอด', 'พอท', 'พอเพ', 'พอแล', 'ภายภาคหน', 'ภายหน', 'ภายหล', 'ภายใต',
'มก', 'มองว', 'มากกว', 'มเต', 'มไปด', 'มไปหมด', 'มๆ', 'ยกให', 'ยง', 'ยงพอ', 'ยง
ว', 'ยงเพ', 'ยงเพราะ', 'ยงแค', 'ยงแต', 'ยงใด', 'ยงไร', 'ยงไหน', 'ยจน', 'ยจนกระท',
'ยจนถ', 'ยด', 'ยน', 'ยนะ', 'ยนแปลง', 'ยบ', 'ยย', 'ยล', 'ยว', 'ยวก', 'ยวข', 'ยว
น', 'ยวเน', 'ยวๆ', 'ยอมร', 'ยเน', 'ยเอง', 'ยแล', 'ยโน', 'รณ', 'รวดเร', 'รวมก', 'รว
มด', 'รวมถ', 'รวมท', 'ระหว', 'วก', 'วง', 'วงก', 'วงต', 'วงถ', 'วงท', 'วงน', 'วงระห
ว', 'วงหน', 'วงหล', 'วงแรก', 'วงๆ', 'วถ', 'วท', 'วน', 'วนจน', 'วนด', 'วนท', 'วน
น', 'วนมาก', 'วนเก', 'วนแต', 'วนใด', 'วนใหญ', 'วม', 'วมก', 'วมด', 'วมม', 'วย',
'วยก', 'วยท', 'วยประการฉะน', 'วยว', 'วยเช', 'วยเพราะ', 'วยเหต', 'วยเหม', 'วเสร', 'ว
แต', 'วๆ', 'สม', 'สำค', 'หมดก', 'หมดส', 'หร', 'หล', 'หากว', 'หากแม', 'หาร', 'หา
ใช', 'อก', 'อค', 'อคร', 'อคราว', 'อคราวก', 'อคราวท', 'อง', 'องจาก', 'องมาจาก', 'อ
จะ', 'อจาก', 'อด', 'อถ', 'อท', 'อน', 'อนก', 'อนข', 'อนมาทาง', 'อนว', 'อนหน',
'อนๆ', 'อบ', 'อบจะ', 'อบๆ', 'อม', 'อมก', 'อมด', 'อมท', 'อมเพ', 'อย', 'อยกว', 'อ
ยคร', 'อยจะ', 'อยเป', 'อยไปทาง', 'อยๆ', 'อว', 'อวาน', 'อาจเป', 'อเก', 'อเช', 'อเป
ล', 'อเม', 'อเย', 'อใด', 'อให', 'อไง', 'อไป', 'อไม', 'อไร', 'อไหร', 'าก', 'าง',
'างก', 'างขวาง', 'างจะ', 'างด', 'างต', 'างท', 'างน', 'างบน', 'างมาก', 'างย', 'าง
ล', 'างละ', 'างหน', 'างหาก', 'างเค', 'างเช', 'างเด', 'างโน', 'างใด', 'างไร', 'างไร
ก', 'างไรเส', 'างไหน', 'างๆ', 'าจะ', 'าท', 'าน', 'านาน', 'านๆ', 'าพเจ', 'าย', 'าย
ก', 'ายว', 'ายใด', 'ายๆ', 'าว', 'าวค', 'าส', 'าหร', 'าหาก', 'าฯ', 'าใจ', 'าใด', 'าใ
ห', 'าไร', 'าไหร', 'าๆ', 'เก', 'เข', 'เฉกเช', 'เช', 'เด', 'เต', 'เถ', 'เท', 'เน', 'เ
ป', 'เปล', 'เผ', 'เพ', 'เพราะฉะน', 'เพราะว', 'เม', 'เร', 'เล', 'เส', 'เสม', 'เสร', 'เ
ห', 'เหต', 'เหล', 'เอ', 'แค', 'แด', 'แต', 'แท', 'แน', 'แม', 'แล', 'แสดงว', 'แ
ห', 'แหล', 'ใกล', 'ใช', 'ใด', 'ใต', 'ในช', 'ในท', 'ในระหว', 'ในเม', 'ให', 'ใหญ',
'ไหม', 'ไข', 'ได', 'ไม', 'ไว', 'ไหม'] not in stop_words.
  warnings.warn(
Training time: 5.0701 seconds
Train Accuracy: 0.7709
Validation Accuracy: 0.6235
Test Accuracy: 0.6335
Number of OOV words: 2798
```

In [32]:

```python
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 numpy as np
import time
```

```python
# Define stopwords
stopwords = list(thai_stopwords())

# Initialize TF-IDF Vectorizer
vectorizer = TfidfVectorizer(
    tokenizer=thai_tokenizer,  # Use default tokenization
    stop_words=stopwords,  # Remove Thai stopwords
    max_features=10000  # Limit vocabulary size to avoid overfitting
)

# Create a logistic regression pipeline
model = Pipeline([
    ("tfidf", vectorizer),
    ("classifier", LogisticRegression(max_iter=1000, random_state=42))
])

# Train model
start = time.time()
model.fit(X_train, y_train)
end = time.time()

# Evaluate model
train_acc = model.score(X_train, y_train)
val_acc = model.score(X_val, y_val)
test_acc = model.score(X_test, 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}")

# Check OOV words (words in test set not seen in train set)
train_vocab = set(vectorizer.get_feature_names_out())
test_vocab = set(word for sentence in X_test for word in sentence.split())
oov_words = test_vocab - train_vocab
print(f"Number of OOV words: {len(oov_words)}")
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:528: U
serWarning: The parameter 'token_pattern' will not be used since 'tokenizer' is n
ot None'
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:409: U
serWarning: Your stop_words may be inconsistent with your preprocessing. Tokenizi
ng the stop words generated tokens ['กระไร', 'กาลนาน', 'ขึ้น', 'ดังที่', 'ดี', 'ดีกว่า',
'ด้อย', 'ตัว', 'ต่อไป', 'ถัดไป', 'ทั่วถึง', 'ทำ', 'ที่จะ', 'ท่าน', 'ท้าย', 'นา', 'บอ', 'บัด
', 'ระยะเวลา', 'ล่ะ', 'วันวาน', 'สม', 'สมบูรณ์', 'สำ', 'หน้า', 'หรับ', 'หา', 'อย', 'เกี่ย
ว', 'เก่า', 'เดี๋ยวนี้', 'เย็น', 'เล่า', 'เสมือน', 'เหมือนกัน', 'แด่', 'แม้น', 'แหล่', 'โง้น',
'โน้น', 'ใด', 'ไว', 'ไหม', '\ufeff'] not in stop_words.
  warnings.warn(
Training time: 3.0621 seconds
Train Accuracy: 0.7650
Validation Accuracy: 0.6939
Test Accuracy: 0.6971
Number of OOV words: 2694
```

1.What tokenizer will you use? Why?

Ans: pythainlp.word_tokenize เพราะ เชื่อ ว่าออกแบบมาเพื่อภาษาไทยโดยเฉพาะ ดังที่เห็นว่าaccuracy ของ validation,test สูงกว่า

2.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: ใช่ ใช้pythainlp.thai_stopwords() เพราะมันเป็นคำที่ไม่สื่อความหมายอะไรอยู่แล้ว

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

Ans: 2694

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 [ ]:

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 [ ]:

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.