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

--2025-02-15 17:31:20--
https://www.dropbox.com/s/37u83g55p19kvrl/clean-phone-data-for-
students.csv
Resolving www.dropbox.com (www.dropbox.com)... 162.125.65.18,
2620:100:6021:18::a27d:4112
Connecting to www.dropbox.com (www.dropbox.com)|162.125.65.18|:443...
connected.
HTTP request sent, awaiting response... 302 Found
Location: https://www.dropbox.com/scl/fi/8h8hvsw9uj6o0524lfe4i/clean-
phone-data-for-students.csv?rlkey=lwv5xbf16jerehnv3lfgq5ue6
[following]
--2025-02-15 17:31:20--
https://www.dropbox.com/scl/fi/8h8hvsw9uj6o0524lfe4i/clean-phone-data-
for-students.csv?rlkey=lwv5xbf16jerehnv3lfgq5ue6
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
```

```
Location:
https://uc776968fa7b8fef321a00b88019.dl.dropboxusercontent.com/cd/0/
inline/
CkKZqgsnCeK_cKn_F_r3AHWaZekPiUFbXPjkJ4ZJHHgogAQO1mChEOUYHrrhbXMe9R9gmH
HmEniUmay7iiBTFp3FHu2zmEi0mfm4AFLanWxwsxFcNGH4IxP-8iqNVDU-Hgw/file#
[following]
--2025-02-15 17:31:21--
https://uc776968fa7b8fef321a00b88019.dl.dropboxusercontent.com/cd/0/
inline/
CkKZqgsnCeK_cKn_F_r3AHWaZekPiUFbXPjkJ4ZJHHgogAQO1mChEOUYHrrhbXMe9R9gmH
HmEniUmay7iiBTFp3FHu2zmEi0mfm4AFLanWxwsxFcNGH4IxP-8iqNVDU-Hgw/file
Resolving uc776968fa7b8fef321a00b88019.dl.dropboxusercontent.com
(uc776968fa7b8fef321a00b88019.dl.dropboxusercontent.com)...
162.125.65.15, 2620:100:6021:15::a27d:410f
Connecting to uc776968fa7b8fef321a00b88019.dl.dropboxusercontent.com
(uc776968fa7b8fef321a00b88019.dl.dropboxusercontent.com)|
162.125.65.15|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2518977 (2.4M) [text/plain]
Saving to: ‘clean-phone-data-for-students.csv’

clean-phone-data-fo 100%[===================>]   2.40M  --.-KB/s    in
0.03s

2025-02-15 17:31:21 (82.2 MB/s) - ‘clean-phone-data-for-students.csv’
saved [2518977/2518977]


!pip install pythainlp

Collecting pythainlp
  Downloading pythainlp-5.0.5-py3-none-any.whl.metadata (7.5 kB)
Requirement already satisfied: requests>=2.22.0 in
/usr/local/lib/python3.10/dist-packages (from pythainlp) (2.32.3)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.22.0-
>pythainlp) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.22.0-
>pythainlp) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.22.0-
>pythainlp) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.22.0-
>pythainlp) (2025.1.31)
Downloading pythainlp-5.0.5-py3-none-any.whl (17.9 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.9/17.9 MB 24.8 MB/s eta
0:00:0000:0100:01
```

Import Libs

```
%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

data_df = pd.read_csv('clean-phone-data-for-students.csv')

def clean_data(df):
    """Cleans the dataset by selecting relevant columns, normalizing
labels,
    trimming whitespace, and removing duplicates."""

    # Select and rename columns
    df = df[["Sentence Utterance",
"Object"]].rename(columns={"Sentence Utterance": "input", "Object":
"raw_label"})

    # Normalize label (lowercase)
    df["clean_label"] = df["raw_label"].str.lower()

    # Trim white spaces in input column
    df["input"] = df["input"].str.strip()

    # Remove duplicates based on input
    df = df.drop_duplicates(subset="input", keep="first")

    # Drop the raw label column
    df.drop(columns=["raw_label"], inplace=True)

    return df

# Apply cleaning function
data_df = clean_data(data_df)

# Display summary
display(data_df.describe())
display(data_df["clean_label"].unique())
```

	input	clean_label
count	13367	13367
unique	13367	26

```
top     สอบถามโปรโมชั่นปัจจุบันที่ใช้อยู่ค่ะ   service
freq                                       1        2108
```

```
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)
```

```
# Mapping and Trimming
data = data_df.to_numpy()
unique_label = data_df.clean_label.unique()

label_2_num_map = dict(zip(unique_label, range(len(unique_label))))
num_2_label_map = dict(zip(range(len(unique_label)), unique_label))

data[:,1] = np.vectorize(label_2_num_map.get)(data[:,1])

def strip_str(string):
    return string.strip()
data[:,0] = np.vectorize(strip_str)(data[:,0])

display(data)
```

```
array([['<PHONE_NUMBER_REMOVED> ผมไปจ่ายเงินที่ Counter Services เค้าเช็ต
3276.25 บาท เมื่อวานที่ผมเช็คที่ศูนย์บอกมียอด 3057.79 บาท',
        0],
       ['internet ยังความเร็วอยู่เท่าไหร ครับ', 1],
       ['ตะกี้ไปชำระค่าบริการไปแล้ว แต่ยังใช้งานไม่ได้ ค่ะ', 2],
       ...,
       ['ยอดเงินเหลือเท่าไหร่ค่ะ', 7],
       ['ยอดเงินในระบบ', 7],
       ['สอบถามโปรโมชั่นปัจจุบันที่ใช้อยู่ค่ะ', 1]], dtype=object)
```

```
# Split
from sklearn.model_selection import train_test_split

# Constants
SEED = 42
MIN_INSTANCES = 10  # Minimum instances per class


def filter_data(data_df, min_instances=MIN_INSTANCES):
    """
    Filters classes with fewer than `min_instances` occurrences.
    Returns filtered input (X) and labels (y).
```

```
    """
    class_counts = data_df["clean_label"].value_counts()
    valid_classes = class_counts[class_counts >= min_instances].index

    filtered_data =
data_df[data_df["clean_label"].isin(valid_classes)]
    return filtered_data["input"],
filtered_data["clean_label"].astype(int)

def split_data(data_df, random_state=SEED,
min_instances=MIN_INSTANCES):
    """
    Splits data into train (80%), validation (10%), and test (10%)
sets.
    Ensures stratification and filtering of rare classes.
    """
    # Filter classes
    X, y = filter_data(data_df, min_instances)

    # Split 80% Train, 20% Temp
    X_train, X_temp, y_train, y_temp = train_test_split(
        X, y, test_size=0.20, stratify=y, random_state=random_state
    )

    # Split 10% Validation, 10% Test
    X_val, X_test, y_val, y_test = train_test_split(
        X_temp, y_temp, test_size=0.50, stratify=y_temp,
random_state=random_state
    )

    print(f"Train size: {len(X_train)}")
    print(f"Validation size: {len(X_val)}")
    print(f"Test size: {len(X_test)}")

    return (
        np.array(X_train), np.array(X_val), np.array(X_test),
        np.array(y_train), np.array(y_val), np.array(y_test)
    )

# Convert to DataFrame
df = pd.DataFrame(data, columns=['input', 'clean_label'])

# Split dataset
X_train, X_val, X_test, y_train, y_val, y_test = split_data(df)

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.

```
print("TfidfVectorizer + Logistic Regression")
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from pythainlp.corpus import thai_stopwords
import time

thai_stopwords_list = list(thai_stopwords())

vectorizer = TfidfVectorizer(
    tokenizer=None,
    stop_words=thai_stopwords_list,
    max_features=5000
)

model = LogisticRegression(random_state=SEED)

text_clf = Pipeline([
    ('tfidf', vectorizer),
    ('clf', model)
])

start_time = time.time()
text_clf.fit(X_train, y_train)
end_time = time.time()
print(f"Training time: {end_time - start_time:.4f} seconds")

y_pred_train = text_clf.predict(X_train)
y_pred_val = text_clf.predict(X_val)
y_pred_test = text_clf.predict(X_test)

train_acc = np.mean(y_train.astype(int) == y_pred_train)
val_acc = np.mean(y_val.astype(int) == y_pred_val)
test_acc = np.mean(y_test.astype(int) == y_pred_test)

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

```
TfidfVectorizer + Logistic Regression

/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/
text.py:409: UserWarning: Your stop_words may be inconsistent with
your preprocessing. Tokenizing the stop words generated tokens ['กคน',
'กคร', 'กครา', 'กคราว', 'กจะ', 'กช', 'กต', 'กท', 'กทาง', 'กน',
'กระท', 'กระน', 'กระไร', 'กล', 'กว', 'กส', 'กหน', 'กอ', 'กอย', 'กำล',
```

'กเม', 'กแห', 'กๆ', 'ขณะท', 'ขณะน', 'ขณะหน', 'ขณะเด', 'คงอย',
'คร', 'ครบคร', 'ครบถ', 'คราท', 'คราน', 'คราวก', 'คราวท', 'คราวน',
'คราวหน', 'คราวหล', 'คราวโน', 'คราหน', 'คล', 'งก', 'งกระน', 'งกล',
'งกว', 'งข', 'งคง', 'งคน', 'งครา', 'งคราว', 'งง', 'งจ', 'งจน', 'งจะ',
'งจาก', 'งต', 'งท', 'งน', 'งบ', 'งปวง', 'งมวล', 'งละ', 'งว', 'งส',
'งหน', 'งหมด', 'งหมาย', 'งหล', 'งหลาย', 'งอย', 'งเก', 'งเคย', 'งเน',
'งเป', 'งเม', 'งแก', 'งแต', 'งแม', 'งแล', 'งโง', 'งโน', 'งใด', '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', 'ซะก',
'ซะจนกระท', 'ซะจนถ', 'ณๆ', 'ดการ', 'ดงาน', 'ดดล', 'ดต', 'ดทำ',
'ดน', 'ดว', 'ดหน', 'ดหา', 'ดเด', 'ดเผย', 'ดแจง', 'ดให', 'ดไป', 'ดๆ',
'ตลอดถ', 'ตลอดท', 'ตลอดป', 'ตลอดว', 'ตามด', 'ตามท', 'ตามแต', 'ทว',
'ทำให', 'นก', 'นการ', 'นกาลนาน', 'นควร', 'นจะ', 'นด', 'นต', 'นท',
'นน', 'นนะ', 'นนาน', 'นมา', 'นมาก', 'นย', 'นยง', 'นยาว', 'นละ',
'นว', 'นวาน', 'นอ', 'นอกจากท', 'นอกจากน', 'นอกจากว', 'นอกน', 'นอก
เหน', 'นอาท', 'นา', 'นเคย', 'นเด', 'นเถอะ', 'นเน', 'นเป', 'นเพ', 'น
เพราะ', 'นเพราะว', 'นเม', 'นเอง', 'นแก', 'นแต', 'นและก', 'นแหละ',
'นใด', 'นใดน', 'นไง', 'นได', 'นไป', 'นไร', 'นไว', 'นไหน', 'นไหม',
'นๆ', 'บจากน', 'บต', 'บรอง', 'บว', 'บอกว', 'บอกแล', 'บางกว', 'บางคร',
'บางท', 'บางแห', 'บเน', 'บแต', 'ปฏ', 'ปร', 'ประการฉะน', 'ประการหน',
'ปรากฏว', 'พบว', 'พร', 'พวกก', 'พวกค', 'พวกฉ', 'พวกท', 'พวกน',
'พวกม', 'พวกโน', 'พอก', 'พอด', 'พอต', 'พอท', 'พอเพ', 'พอแล', 'ภาย
ภาคหน', 'ภายหน', 'ภายหล', 'ภายใต', 'มก', 'มองว', 'มากว', 'มเต',
'มไปด', 'มไปหมด', 'มๆ', 'ยกให', 'ยง', 'ยงพอ', 'ยงว', 'ยงเพ', 'ยง
เพราะ', 'ยงแค', 'ยงแต', 'ยงใด', 'ยงไร', 'ยงไหน', 'ยจน', 'ยจนกระท',
'ยจนถ', 'ยด', 'ยน', 'ยนะ', 'ยนแปลง', 'ยบ', 'ยย', 'ยล', 'ยว', 'ยวก',
'ยวข', 'ยวน', 'ยวเน', 'ยวๆ', 'ยอมร', 'ยเน', 'ยเอง', 'ยแล', 'ยโน',
'รณ', 'รวดเร', 'รวมก', 'รวมด', 'รวมถ', 'รวมท', 'ระหว', 'วก', 'วง',
'วงก', 'วงต', 'วงถ', 'วงท', 'วงน', 'วงระหว', 'วงหน', 'วงหล', 'วงแรก',
'วงๆ', 'วถ', 'วท', 'วน', 'วนจน', 'วนด', 'วนท', 'วนน', 'วนมาก', 'วนเก',
'วนแต', 'วนใด', 'วนใหญ', 'วม', 'วมก', 'วมด', 'วมม', 'วย', 'วยก',
'วยท', 'วยประการฉะน', 'วยว', 'วยเช', 'วยเพราะ', 'วยเหต', 'วยเหม',
'วเสร', 'วแต', 'วๆ', 'สม', 'สำค', 'หมดก', 'หมดส', 'หร', 'หล', 'หากว',
'หากแม', 'หาร', 'หาใช', 'อก', 'อค', 'อคร', 'อคราว', 'อคราวก', 'อคราว
ท', 'อง', 'องจาก', 'องมาจาก', 'อจะ', 'อจาก', 'อด', 'อถ', 'อท', 'อน',
'อนก', 'อนข', 'อนมาทาง', 'อนว', 'อนหน', 'อนๆ', 'อบ', 'อบจะ', 'อบๆ',
'อม', 'อมก', 'อมด', 'อมท', 'อมเพ', 'อย', 'อยกว', 'อยคร', 'อยจะ',
'อยเป', 'อยไปทาง', 'อยๆ', 'อว', 'อวาน', 'อาจเป', 'อเก', 'อเช', 'อเปล',
'อเม', 'อเย', 'อใด', 'อให', 'อไง', 'อไป', 'อไม', 'อไร', 'อไหร', 'าก',
'าง', 'างก', 'างขวาง', 'างจะ', 'างด', 'างต', 'างท', 'างน', 'างบน', 'า
งมาก', 'างย', 'างล', 'างละ', 'างหน', 'างหาก', 'างเค', 'างเช', 'างเด',
'างโน', 'างใด', 'างไร', 'างไรก', 'างไรเส', 'างไหน', 'างๆ', 'าจะ', 'าท',
'าน', 'านาน', 'านๆ', 'าพเจ', 'าย', 'ายก', 'ายว', 'ายใด', 'ายๆ', 'าว',
'าวค', 'าส', 'าหร', 'าหาก', 'าฯ', 'าใจ', 'าใด', 'าให', 'าไร', 'าไหร',

```
'าๆ', 'เก', 'เข', 'เฉกเช่', 'เช', 'เด', 'เต', 'เถ', 'เท', 'เน', 'เป',
'เปล', 'เผ', 'เพ', 'เพราะฉะนั้น', 'เพราะว่า', 'เม', 'เร', 'เล', 'เส', 'เสม',
'เสร', 'เห', 'เหต', 'เหล', 'เอ', 'แค', 'แด', 'แต', 'แท', 'แน', 'แม',
'แล', 'แสดงว่า', 'แห', 'แหล', 'ใกล', 'ใช', 'ใด', 'ใต', 'ในช', 'ในท',
'ในระหว่', 'ในเม', 'ให', 'ใหญ่', 'ใหม', 'ไข', 'ได', 'ไม', 'ไว', 'ไหม']
not in stop_words.
  warnings.warn(

Training time: 2.2299 seconds
Train Accuracy: 0.7574
Validation Accuracy: 0.6257
Test Accuracy: 0.6178

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/
_logistic.py:458: ConvergenceWarning: 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(
```

```python
print("TfidfVectorizer + Logistic Regression +
pythainlp.word_tokenize")
from pythainlp.tokenize import word_tokenize

thai_stopwords_list = list(thai_stopwords())


def thai_tokenizer(text):
    return word_tokenize(text, keep_whitespace=False)


vectorizer = TfidfVectorizer(
    tokenizer=thai_tokenizer,
    stop_words=thai_stopwords_list,
    max_features=5000,
)

model = LogisticRegression(random_state=SEED)

text_clf = Pipeline([("tfidf", vectorizer), ("clf", model)])

start_time = time.time()
text_clf.fit(X_train, y_train)
```

```python
end_time = time.time()
print(f"Training time: {end_time - start_time:.4f} seconds")

y_pred_train = text_clf.predict(X_train)
y_pred_val = text_clf.predict(X_val)
y_pred_test = text_clf.predict(X_test)

train_acc = np.mean(y_train.astype(int) == y_pred_train)
val_acc = np.mean(y_val.astype(int) == y_pred_val)
test_acc = np.mean(y_test.astype(int) == y_pred_test)

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

```
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/
text.py:528: UserWarning: The parameter 'token_pattern' will not be
used since 'tokenizer' is not None'
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/
text.py:409: UserWarning: Your stop_words may be inconsistent with
your preprocessing. Tokenizing the stop words generated tokens
['กระไร', 'กาลนาน', 'ซิน', 'ดังที่', 'ดี', 'ดีกว่า', 'ด้อย', 'ตัว', 'ต่อไป',
'ถัดไป', 'ทั่วถึง', 'ทำ', 'ที่จะ', 'ท่าน', 'ท้าย', 'นา', 'บอ', 'บัด', 'ระยะ
เวลา', 'ล่ะ', 'วันวาน', 'สม', 'สมบูรณ์', 'สำ', 'หน้า', 'หรับ', 'หา', 'อย',
'เกี่ยว', 'เก่า', 'เดี๋ยวนี้', 'เย็น', 'เล่า', 'เสมือน', 'เหมือนกัน', 'แด่', 'แม้น',
'แหล่', 'โง้น', 'โน้น', 'ใด', 'ไว', 'ไหม', '\ufeff'] not in stop_words.
  warnings.warn(

TfidfVectorizer + Logistic Regression + pythainlp.word_tokenize

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/
_logistic.py:458: ConvergenceWarning: 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(

Training time: 2.8185 seconds
Train Accuracy: 0.7675
Validation Accuracy: 0.6886
Test Accuracy: 0.6933
```

```python
def calculate_oov_words(X_train, X_test):
    """
    Calculates the number of out-of-vocabulary (OOV) words in the test
set
    compared to the training set.

    Parameters:
        X_train (list): List of training texts.
        X_test (list): List of test texts.

    Returns:
        int: Number of OOV words.
    """
    train_words = {word for text in X_train for word in text.split()}
    test_words = {word for text in X_test for word in text.split()}

    oov_words = test_words - train_words
    oov_count = len(oov_words)

    return oov_count

# Compute OOV words
oov_count = calculate_oov_words(X_train, X_test)
print(oov_count)
```

```
2245
```

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 and why?

A: I will use pythainlp.word_tokenize because it is specifically designed for processing Thai text. The results show that it provides:

- Higher accuracy across all datasets.
- Better generalization, as indicated by improved validation accuracy (val_acc).

Will you remove stopwords (e.g., "a," "an," "the," "to" in English) in your TF-IDF process? Is it necessary?

A: Yes, I will remove Thai stopwords using pythainlp.thai_stopwords(). Eliminating these common but non-informative words helps enhance model efficiency by reducing noise and improving classification performance.

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

A: 2245

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.