NLP Assignment 1 (40% of grade): Text classification for Fake News Detection

This coursework will involve you implementing functions for a text classifier, which you will train to detect **fake news** in a corpus of approx. 10,000 statements, which will be split into a 80%/20% training/test split.

In this template you are given the basis for that implementation, though some of the functions are missing, which you have to fill in.

Follow the instructions file **NLP_Assignment_1_Instructions.pdf** for details of each question - the outline of what needs to be achieved for each question is as below.

You must submit all **ipython notebooks and extra resources you need to run the code if you've added them** in the code submission, and a **2 page report (pdf)** in the report submission on QMPlus where you report your methods and findings according to the instructions file for each question.

In [1]:

```
!pip install nltk
```

```
Requirement already satisfied: nltk in /opt/conda/lib/python3.10/site-pa
ckages (3.7)
Requirement already satisfied: tqdm in /opt/conda/lib/python3.10/site-pa
ckages (from nltk) (4.64.1)
Requirement already satisfied: click in /opt/conda/lib/python3.10/site-p
ackages (from nltk) (8.1.3)
Requirement already satisfied: regex>=2021.8.3 in /opt/conda/lib/python3
.10/site-packages (from nltk) (2022.10.31)
Requirement already satisfied: joblib in /opt/conda/lib/python3.10/site-
packages (from nltk) (1.2.0)

[notice] A new release of pip available: 22.3 -> 22.3.1
[notice] To update, run: pip install --upgrade pip
```

In [2]:

```
import pandas  as pd
import csv                          # csv reader
from sklearn.svm import LinearSVC
from nltk.classify import SklearnClassifier
from sklearn.pipeline import Pipeline
from sklearn.metrics import precision_recall_fscore_support # to report o
import numpy as np
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.classify import accuracy
```

Question 1: Input and Basic preprocessing (10 marks)

In [3]:
```
def convert_label(label):
    """Converts the multiple classes into two,
    making it a binary distinction between fake news and real."""
    #return label
    # Converting the multiclass labels to binary label
    labels_map = {
        'true': 'REAL',
        'mostly-true': 'REAL',
        'half-true': 'REAL',
        'false': 'FAKE',
        'barely-true': 'FAKE',
        'pants-fire': 'FAKE'
    }
    return labels_map[label]


def parse_data_line(data_line):
    # Should return a tuple of the label as just FAKE or REAL and the sta
    # e.g. (label, statement)
    data_line[0]=convert_label(data_line[0])
    a=(data_line[0],data_line[1])
    return a
```

In [4]:
```
# Input: a string of one statement
def pre_process(text):
    # Should return a list of tokens
    # DESCRIBE YOUR METHOD IN WORDS
    a=nltk.word_tokenize(text)
    interpunctuations = [',', '.', ':', ';', '?', '(', ')', '[', ']', '&'
    cutwords2 = [word for word in a if word not in interpunctuations]
    stops = set(stopwords.words("english"))
    cutwords3 = [word for word in cutwords2 if word not in stops]
    cutwords4 = []
    for cutword in cutwords3:
        cutwords4.append(PorterStemmer().stem(cutword))
    return cutwords4
```

Question 2: Basic Feature Extraction (20 marks)

In [5]:
```
def to_feature_vector(tokens):
    global_feature_dict = {}
    # Should return a dictionary containing features as keys, and weights
    # DESCRIBE YOUR METHOD IN WORDS
    for item in tokens:
        global_feature_dict[item]=global_feature_dict.get(item,0)+1
    return global_feature_dict
```

In [6]:
```
def load_data(path):
    """Load data from a tab-separated file and append it to raw_data."""
    raw_data=[]
    reader = pd.read_csv(path,delimiter='\t')
    for i in range(len(reader)):
            (label, text) = parse_data_line(reader.iloc[i,:])
            raw_data.append((text, label))
    return raw_data

def split_and_preprocess_data(percentage,path):
    """Split the data between train_data and test_data according to the p
    and performs the preprocessing."""
    train_data=[]
    test_data=[]
    raw_data=load_data(path)
    num_samples = len(raw_data)
    num_training_samples = int((percentage * num_samples))
    for (text, label) in raw_data[:num_training_samples]:
        train_data.append((to_feature_vector(pre_process(text)),label))
    for (text, label) in raw_data[num_training_samples:]:
        test_data.append((to_feature_vector(pre_process(text)),label))
    return train_data ,test_data
```

In [7]:
```
# TRAINING AND VALIDATING OUR CLASSIFIER

def train_classifier(data):
    print("Training Classifier...")
    pipeline =  Pipeline([('svc', LinearSVC())])
    model=SklearnClassifier(pipeline).train(data)
    return model
```

In [8]:
```
train_data ,test_data=split_and_preprocess_data(0.9,'fake_news.tsv')
```

```
/tmp/ipykernel_258/909790341.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-d
ocs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data_line[0]=convert_label(data_line[0])
------------------------------------------------------------------------
---
LookupError                               Traceback (most recent call la
st)
Cell In [8], line 1
```

```
----> 1 train_data ,test_data=split_and_preprocess_data(0.9,'fake_news.t
sv')

Cell In [6], line 19, in split_and_preprocess_data(percentage, path)
     17 num_training_samples = int((percentage * num_samples))
     18 for (text, label) in raw_data[:num_training_samples]:
---> 19     train_data.append((to_feature_vector(pre_process(text)),labe
l))
     20 for (text, label) in raw_data[num_training_samples:]:
     21     test_data.append((to_feature_vector(pre_process(text)),label
))

Cell In [4], line 5, in pre_process(text)
      2 def pre_process(text):
      3     # Should return a list of tokens
      4     # DESCRIBE YOUR METHOD IN WORDS
----> 5     a=nltk.word_tokenize(text)
      6     interpunctuations = [',', '.', ':', ';', '?', '(', ')', '[',
']', '&', '!', '*', '@', '#', '$', '%']
      7     cutwords2 = [word for word in a if word not in
interpunctuations]

File /opt/conda/lib/python3.10/site-packages/nltk/tokenize/__init__.py:1
29, in word_tokenize(text, language, preserve_line)
    114 def word_tokenize(text, language="english", preserve_line=False)
:
    115     """
    116     Return a tokenized copy of *text*,
    117     using NLTK's recommended word tokenizer
   (...)
    127     :type preserve_line: bool
    128     """
--> 129     sentences = [text] if preserve_line else sent_tokenize(text,
language)
    130     return [
    131         token for sent in sentences for token in
_treebank_word_tokenizer.tokenize(sent)
    132     ]

File /opt/conda/lib/python3.10/site-packages/nltk/tokenize/__init__.py:1
06, in sent_tokenize(text, language)
     96 def sent_tokenize(text, language="english"):
     97     """
     98     Return a sentence-tokenized copy of *text*,
     99     using NLTK's recommended sentence tokenizer
   (...)
    104     :param language: the model name in the Punkt corpus
    105     """
--> 106     tokenizer = load(f"tokenizers/punkt/{language}.pickle")
    107     return tokenizer.tokenize(text)

File /opt/conda/lib/python3.10/site-packages/nltk/data.py:750, in load(r
esource_url, format, cache, verbose, logic_parser, fstruct_reader, encod
ing)
    747     print(f"<<Loading {resource_url}>>")
    749 # Load the resource.
--> 750 opened_resource = _open(resource_url)
    752 if format == "raw":
    753     resource_val = opened_resource.read()

File /opt/conda/lib/python3.10/site-packages/nltk/data.py:876, in _open(
resource_url)
    873 protocol, path_ = split_resource_url(resource_url)
    875 if protocol is None or protocol.lower() == "nltk":
--> 876     return find(path_, path + [""]).open()
    877 elif protocol.lower() == "file":
    878     # urllib might not use mode='rb', so handle this one ourselv
es:
    879     return find(path_, [""]).open()

File /opt/conda/lib/python3.10/site-packages/nltk/data.py:583, in find(r
esource_name, paths)
    581 sep = "*" * 70
    582 resource_not_found = f"\n{sep}\n{msg}\n{sep}\n"
--> 583 raise LookupError(resource_not_found)

LookupError: 
**********************************************************************
  Resource punkt not found.
  Please use the NLTK Downloader to obtain the resource:

  >>> import nltk
  >>> nltk.download('punkt')
  
  For more information see: https://www.nltk.org/data.html

  Attempted to load tokenizers/punkt/PY3/english.pickle

  Searched in:
    - '/home/jovyan/nltk_data'
    - '/opt/conda/nltk_data'
    - '/opt/conda/share/nltk_data'
    - '/opt/conda/lib/nltk_data'
    - '/usr/share/nltk_data'
    - '/usr/local/share/nltk_data'
    - '/usr/lib/nltk_data'
    - '/usr/local/lib/nltk_data'
    - ''
**********************************************************************
```

Question 3: Cross-validation (20 marks)

In [9]:
```
#solution
from sklearn.metrics import classification_report


def cross_validate(dataset, folds):
    results = []
    fold_size = int(len(dataset)/folds) + 1

    for i in range(0,len(dataset),int(fold_size)):
        # insert code here that trains and tests on the 10 folds of data
        result=train_classifier(dataset[i:i+fold_size])
        results.append(accuracy(result, test_data))
        print("Fold start on items %d - %d" % (i, i+fold_size))
        # FILL IN THE METHOD HERE

    return results
```

In [10]:
```
# PREDICTING LABELS GIVEN A CLASSIFIER

def predict_labels(samples, classifier):
    """Assuming preprocessed samples, return their predicted labels from
    return classifier.classify_many(samples)

def predict_label_from_raw(sample, classifier):
    """Assuming raw text, return its predicted label from the classifier
    return classifier.classify(to_feature_vector(pre_process(sample)))
```

In [11]:
```
# MAIN

# loading reviews
# initialize global lists that will be appended to by the methods below
raw_data = []          # the filtered data from the dataset file
train_data = []        # the pre-processed training data as a percentage
test_data = []         # the pre-processed test data as a percentage of t


# references to the data files
data_file_path = 'fake_news.tsv'

# Do the actual stuff (i.e. call the functions we've made)
# We parse the dataset and put it in a raw data list
print("Now %d rawData, %d trainData, %d testData" % (len(raw_data), len(t
      "Preparing the dataset...",sep='\n')

raw_data=load_data(data_file_path)

# We split the raw dataset into a set of training data and a set of test
# You do the cross validation on the 80% (training data)
# We print the number of training samples and the number of features befo
print("Now %d rawData, %d trainData, %d testData" % (len(raw_data), len(t
      "Preparing training and test data...",sep='\n')

train_data,test_data=split_and_preprocess_data(0.8,data_file_path)
global_feature_dict=to_feature_vector(train_data[0][0])

# We print the number of training samples and the number of features afte
print("After split, %d rawData, %d trainData, %d testData" % (len(raw_dat
      "Training Samples: ", len(train_data), "Features: ", len(global_fea
```

```
Now 0 rawData, 0 trainData, 0 testData
Preparing the dataset...
/tmp/ipykernel_258/909790341.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-d
ocs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data_line[0]=convert_label(data_line[0])
Now 10241 rawData, 0 trainData, 0 testData
Preparing training and test data...
/tmp/ipykernel_258/909790341.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-d
ocs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data_line[0]=convert_label(data_line[0])
-------------------------------------------------------------------------
---
LookupError                               Traceback (most recent call la
st)
```

```
Cell In [11], line 26
     20 # We split the raw dataset into a set of training data and a set
of test data (80/20)
     21 # You do the cross validation on the 80% (training data)
     22 # We print the number of training samples and the number of feat
ures before the split
     23 print("Now %d rawData, %d trainData, %d testData" % (len(raw_dat
a), len(train_data), len(test_data)),
     24       "Preparing training and test data...",sep='\n')
---> 26 train_data,test_data=split_and_preprocess_data(0.8,data_file_pat
h)
     27 global_feature_dict=to_feature_vector(train_data[0][0])
     29 # We print the number of training samples and the number of feat
ures after the split

Cell In [6], line 19, in split_and_preprocess_data(percentage, path)
     17 num_training_samples = int((percentage * num_samples))
     18 for (text, label) in raw_data[:num_training_samples]:
---> 19     train_data.append((to_feature_vector(pre_process(text)),labe
l))
     20 for (text, label) in raw_data[num_training_samples:]:
     21     test_data.append((to_feature_vector(pre_process(text)),label
))

Cell In [4], line 5, in pre_process(text)
      2 def pre_process(text):
      3     # Should return a list of tokens
      4     # DESCRIBE YOUR METHOD IN WORDS
----> 5     a=nltk.word_tokenize(text)
      6     interpunctuations = [',', '.', ':', ';', '?', '(', ')', '[',
']', '&', '!', '*', '@', '#', '$', '%']
      7     cutwords2 = [word for word in a if word not in
interpunctuations]

File /opt/conda/lib/python3.10/site-packages/nltk/tokenize/__init__.py:1
29, in word_tokenize(text, language, preserve_line)
    114 def word_tokenize(text, language="english", preserve_line=False)
:
    115     """
    116     Return a tokenized copy of *text*,
    117     using NLTK's recommended word tokenizer
   (...)
    127     :type preserve_line: bool
    128     """
--> 129     sentences = [text] if preserve_line else sent_tokenize(text,
language)
    130     return [
    131         token for sent in sentences for token in
_treebank_word_tokenizer.tokenize(sent)
    132     ]

File /opt/conda/lib/python3.10/site-packages/nltk/tokenize/__init__.py:1
06, in sent_tokenize(text, language)
```

```
     96 def sent_tokenize(text, language="english"):
     97     """
     98     Return a sentence-tokenized copy of *text*,
     99     using NLTK's recommended sentence tokenizer
   (...)
    104     :param language: the model name in the Punkt corpus
    105     """
--> 106     tokenizer = load(f"tokenizers/punkt/{language}.pickle")
    107     return tokenizer.tokenize(text)

File /opt/conda/lib/python3.10/site-packages/nltk/data.py:750, in load(r
esource_url, format, cache, verbose, logic_parser, fstruct_reader, encod
ing)
    747     print(f"<<Loading {resource_url}>>")
    749 # Load the resource.
--> 750 opened_resource = _open(resource_url)
    752 if format == "raw":
    753     resource_val = opened_resource.read()

File /opt/conda/lib/python3.10/site-packages/nltk/data.py:876, in _open(
resource_url)
    873 protocol, path_ = split_resource_url(resource_url)
    875 if protocol is None or protocol.lower() == "nltk":
--> 876     return find(path_, path + [""]).open()
    877 elif protocol.lower() == "file":
    878     # urllib might not use mode='rb', so handle this one ourselv
es:
    879     return find(path_, [""]).open()

File /opt/conda/lib/python3.10/site-packages/nltk/data.py:583, in find(r
esource_name, paths)
    581 sep = "*" * 70
    582 resource_not_found = f"\n{sep}\n{msg}\n{sep}\n"
--> 583 raise LookupError(resource_not_found)

LookupError: 
**********************************************************************
  Resource punkt not found.
  Please use the NLTK Downloader to obtain the resource:

  >>> import nltk
  >>> nltk.download('punkt')
  
  For more information see: https://www.nltk.org/data.html

  Attempted to load tokenizers/punkt/PY3/english.pickle

  Searched in:
    - '/home/jovyan/nltk_data'
    - '/opt/conda/nltk_data'
    - '/opt/conda/share/nltk_data'
    - '/opt/conda/lib/nltk_data'
    - '/usr/share/nltk_data'
    - '/usr/local/share/nltk_data'
    - '/usr/lib/nltk_data'
    - '/usr/local/lib/nltk_data'
    - ''
**********************************************************************
```

In [12]:
```
results=cross_validate(train_data, 10)  # will work and output overall pe
results
```

Out[12]: `[]`

4. Error Analysis (10 marks)

In [13]:
```
from sklearn import metrics
import matplotlib.pyplot as plt
# a function to make the confusion matrix readable and pretty
def confusion_matrix_heatmap(y_test, preds, labels):
    """Function to plot a confusion matrix"""
    # pass labels to the confusion matrix function to ensure right order
    cm = metrics.confusion_matrix(y_test, preds, labels=labels)
    fig = plt.figure(figsize=(10,10))
    ax = fig.add_subplot(111)
    cax = ax.matshow(cm)
    plt.title('Confusion matrix of the classifier')
    fig.colorbar(cax)
    ax.set_xticks(np.arange(len(labels)))
    ax.set_yticks(np.arange(len(labels)))
    ax.set_xticklabels( labels, rotation=45)
    ax.set_yticklabels( labels)

    for i in range(len(cm)):
        for j in range(len(cm)):
            text = ax.text(j, i, cm[i, j],
                           ha="center", va="center", color="w")

    plt.xlabel('Predicted')
    plt.ylabel('True')

    # fix for mpl bug that cuts off top/bottom of seaborn viz:
    b, t = plt.ylim() # discover the values for bottom and top
    b += 0.5 # Add 0.5 to the bottom
    t -= 0.5 # Subtract 0.5 from the top
    plt.ylim(b, t) # update the ylim(bottom, top) values
    plt.show() # ta-da!
    plt.show()
```

In [14]:
```
preds=[]
for i in range(len(test_data)):
    pred=predict_labels(test_data[i][0],a)
    preds.append(pred)
y_test=np.array(test_data)[:,1]
y_test
preds
```

```
---------------------------------------------------------------------------
---
IndexError                                Traceback (most recent call la
st)
Cell In [14], line 5
      3     pred=predict_labels(test_data[i][0],a)
      4     preds.append(pred)
----> 5 y_test=np.array(test_data)[:,1]
      6 y_test
      7 preds

IndexError: too many indices for array: array is 1-dimensional, but 2 we
re indexed
```

In [15]:
```
confusion_matrix_heatmap(preds,y_test,['REAL','FAKE'])
```

```
---------------------------------------------------------------------------
---
NameError                                 Traceback (most recent call la
st)
Cell In [15], line 1
----> 1 confusion_matrix_heatmap(preds,y_test,['REAL','FAKE'])

NameError: name 'y_test' is not defined
```

Questions 5 (20%) and 6 (20%) (recommend starting a new notebook)

In [16]:
```
# Finally, check the accuracy of your classifier by training on all the t
# and testing on the test set
# Will only work once all functions are complete
functions_complete = True  # set to True once you're happy with your meth
if functions_complete:
    print(test_data[0])   # have a look at the first test data instance
    classifier = train_classifier(train_data)  # train the classifier
    test_true = [t[1] for t in test_data]   # get the ground-truth labels
    test_pred = predict_labels([x[0] for x in test_data], classifier)  #
    final_scores = precision_recall_fscore_support(test_true, test_pred,
    print("Done training!")
    print("Precision: %f\nRecall: %f\nF Score:%f" % final_scores[:3])
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In [16], line 6
      4 functions_complete = True  # set to True once you're happy with your methods for cross val
      5 if functions_complete:
----> 6     print(test_data[0])   # have a look at the first test data instance
      7     classifier = train_classifier(train_data)  # train the classifier
      8     test_true = [t[1] for t in test_data]   # get the ground-truth labels from the data

IndexError: list index out of range
```

```
In [ ]:
```