# lab4-1

October 1, 2023

[31]:
```
#import necessary libraries
import pandas as pd
import string
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
nltk.download('stopwords')
nltk.download('punkt')

from nltk.tokenize import word_tokenize

from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix,␣
 ↪classification_report
from sklearn.metrics import accuracy_score, confusion_matrix,␣
 ↪classification_report
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/hudsonshimanyula/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/hudsonshimanyula/nltk_data…
[nltk_data]   Package punkt is already up-to-date!
```

[5]:
```
#load data set
legal_sentences_training_dir = './dataSets/
 ↪Legal_Sentences_For_Training_With_BERT_With_Label.xlsx'
legal_sentences_testing_dir = './dataSets/Testing_Set_Legal_Sentences.xlsx'


#display training data as df
training_df = pd.read_excel(legal_sentences_training_dir)
testing_df = pd.read_excel(legal_sentences_testing_dir)


print(training_df.head())
```

```
print(testing_df.head())
```

```
   ID                                             Phrase  Sentiment  Label
0   1                    Getting nowhere with surplusage         -1      0
1   2  But the Court nowhere suggested that it would …         -1      0
2   3  Petitioners objection to shaving his beard cla…         -1      0
3   4           That result clashes with everything else         -1      0
4   5  the tolerable duration of police inquiries in …          0      1
   ID                                           sentence  label
0   1  has done nothing to satisfy the probable-cause…      0
1   2  Addressing that question here , the CCA referr…      1
2   3                          standards and procedures      1
3   4  has no comprehension of why he has been single…      0
4   5  an expert , Dr. Woods , who offered the opinio…      2
```

[12]:
```
#Preprocess the text data (remove stopwords, punctuation, lowercase, etc.).
def preprocess_text(text):

    # Remove punctuation
    text = text.translate(str.maketrans('', '', string.punctuation))

    # Convert words to lower case and split them
    text = text.lower()

    # Remove stop words
    stop_words = set(stopwords.words('english'))
    word_tokens = word_tokenize(text)
    filtered_text = [word for word in word_tokens if word not in stop_words]

    # Remove words with length less than 3
    filtered_text = [word for word in filtered_text if len(word) >= 3]

    # Join all
    text = " ".join(filtered_text)

    return text
```

[13]:
```
#clean the training sentences
cleaned_training_df = training_df['Phrase'].apply(preprocess_text)
cleaned_testing_df = testing_df['sentence'].apply(preprocess_text)

training_df['Cleaned_Phrase'] = cleaned_training_df
testing_df['Cleaned_Phrase'] = cleaned_testing_df

#Display only the first 5 rows of the cleaned data
print(training_df[['Phrase', 'Cleaned_Phrase']].head())
print(testing_df[['sentence', 'Cleaned_Phrase']].head())
```

```
                                                  Phrase  \
0                        Getting nowhere with surplusage
1    But the Court nowhere suggested that it would …
2    Petitioners objection to shaving his beard cla…
3              That result clashes with everything else
4    the tolerable duration of police inquiries in …
..                                                   …
571  the driver is not willing to become an affiant…
572  a committee of Congress would be receptive to …
573  to develop a factory survey program that is pr…
574  But it is far from obvious that the residual c…
575  Congress spoke with conspicuous clarity in mak…

                                       Cleaned_Phrase
0                             getting nowhere surplusage
1    court nowhere suggested would narrow bivens ex…
2    petitioners objection shaving beard clashes ar…
3                         result clashes everything else
4    tolerable duration police inquiries trafficsto…
..                                                   …
571                        driver willing become affiant
572  committee congress would receptive suggestion …
573  develop factory survey program predictably rel…
574  far obvious residual clause implicates twin co…
575  congress spoke conspicuous clarity making subs…

[576 rows x 2 columns]
                                                 sentence  \
0    has done nothing to satisfy the probable-cause…
1    Addressing that question here , the CCA referr…
2                              standards and procedures
3    has no comprehension of why he has been single…
4    an expert , Dr. Woods , who offered the opinio…
..                                                   …
495  Assessing Adaptive Functioning in Death Penalt…
496  The alleged bribe took the form of an all-expe…
497  laws that fix the permissible sentences for cr…
498  The BB gun closely resembled a small caliber r…
499  some courts have proceeded to resentence defen…

                                       Cleaned_Phrase
0       done nothing satisfy probablecause requirement
1    addressing question cca referred moore educati…
2                                  standards procedures
3                                comprehension singled
4    expert woods offered opinion mcwilliams suffer…
..                                                   …
495  assessing adaptive functioning death penalty c…
```

```
496  alleged bribe took form allexpensespaid trip l…
497  laws fix permissible sentences criminal offens…
498          gun closely resembled small caliber rifle
499  courts proceeded resentence defendants whose s…

[500 rows x 2 columns]
```

[25]:
```
print(cleaned_training_df.head())

#Perform keyword extraction using TF-IDF
from sklearn.feature_extraction.text import TfidfVectorizer

# Define vectorizer parameters
tfidf_vectorizer = TfidfVectorizer(max_df=1.0, max_features=200000,
                                   min_df=0.0, stop_words='english',
                                   use_idf=True, tokenizer=word_tokenize,␣
 ↪ngram_range=(1,3))

# Fit the vectorizer to training data
tfidf_matrix = tfidf_vectorizer.fit_transform(cleaned_training_df)

# Print the TF-IDF scores
print(tfidf_matrix)

# Get the feature names (words/terms)
feature_names = tfidf_vectorizer.get_feature_names_out()

# Print the first 10 features
print(feature_names[:10])
```

```
0                          getting nowhere surplusage
1    court nowhere suggested would narrow bivens ex…
2    petitioners objection shaving beard clashes ar…
3                      result clashes everything else
4    tolerable duration police inquiries trafficsto…
Name: Phrase, dtype: object
  (0, 3354)     0.5973145577883097
  (0, 7413)     0.5973145577883097
  (0, 3351)     0.5351921506415354
  (1, 823)      0.2677191242715201
  (1, 4919)     0.2677191242715201
  (1, 7368)     0.2677191242715201
  (1, 1824)     0.2677191242715201
  (1, 2929)     0.2677191242715201
  (1, 822)      0.2677191242715201
  (1, 4918)     0.2677191242715201
  (1, 7367)     0.2677191242715201
  (1, 1823)     0.2677191242715201
```

```
  (1, 6750)     0.25143169292387396
  (1, 2928)     0.2677191242715201
  (1, 821)      0.2677191242715201
  (1, 4917)     0.2677191242715201
  (1, 7364)     0.23987557647560742
  (1, 1749)     0.13841769249418262
  (2, 1699)     0.1949495842462718
  (2, 2159)     0.1949495842462718
  (2, 540)      0.1949495842462718
  (2, 1208)     0.1949495842462718
  (2, 782)      0.1949495842462718
  (2, 6924)     0.1949495842462718
  (2, 5049)     0.1949495842462718
  :     :
  (574, 3088)   0.22101509858441615
  (574, 1394)   0.22101509858441615
  (574, 7749)   0.22101509858441615
  (574, 3764)   0.22101509858441615
  (574, 3085)   0.2108569670204692
  (574, 5089)   0.20297770804343987
  (574, 7956)   0.2108569670204692
  (574, 6448)   0.2108569670204692
  (574, 1209)   0.16923834122532622
  (574, 6447)   0.2108569670204692
  (575, 1205)   0.2682277286273054
  (575, 1535)   0.2682277286273054
  (575, 7105)   0.2682277286273054
  (575, 1484)   0.2682277286273054
  (575, 4659)   0.2682277286273054
  (575, 1204)   0.2682277286273054
  (575, 7104)   0.2682277286273054
  (575, 1483)   0.2682277286273054
  (575, 7103)   0.2682277286273054
  (575, 1533)   0.2519093549308431
  (575, 1201)   0.2519093549308431
  (575, 1532)   0.2519093549308431
  (575, 7309)   0.2519093549308431
  (575, 4654)   0.2313506354364935
  (575, 1470)   0.21243484044611569
['016' '016 higher' '016 higher spend' '109a' '17yearold'
 '17yearold victim' '17yearold victim qualify' '1934' '1934 outbreak'
 '1934 outbreak bank']

/Users/hudsonshimanyula/anaconda3/envs/AI_MASTERS_ENV/lib/python3.11/site-
packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter
'token_pattern' will not be used since 'tokenizer' is not None'
  warnings.warn(
```

[26]:
```
#Convert the rating scale into sentiment labels, such as "positive,"␣
↪"negative," and "neutral."
# Convert the ratings into sentiment labels
def to_sentiment(rating):
    rating = int(rating)
    if rating == 1:
        return "negative"
    elif rating == 2:
        return "neutral"
    else:
        return "positive"
```

[28]:
```
#Label as positive, negative, or neutral on training data
training_df['Sentiment_Label'] = training_df['Sentiment'].map({-1: 'negative',␣
↪0: 'neutral', 1: 'positive'})

#Print the first 10 rows
print(training_df[['Phrase', 'Sentiment', 'Sentiment_Label']].head(10))
```

```
                                              Phrase  Sentiment  \
0                    Getting nowhere with surplusage         -1
1  But the Court nowhere suggested that it would …         -1
2  Petitioners objection to shaving his beard cla…         -1
3           That result clashes with everything else         -1
4  the tolerable duration of police inquiries in …          0
5  retrial be tolerable if the trial error could …          0
6                      I would be inclined to agree.          0
7  the trial court was inclined to accept the pro…          1
8  a plaintiff could overcome these hurdles where…          1
9  the procedural hurdles it could impose before …         -1

  Sentiment_Label
0        negative
1        negative
2        negative
3        negative
4         neutral
5         neutral
6         neutral
7        positive
8        positive
9        negative
```

[ ]:
```
#Split the data into training and testing sets.
# Split the data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(tfidf_matrix,␣
↪training_df['Sentiment'], test_size=0.2, random_state=0)
```

[ ]:
```
#Model Training/Buidling
# Train the model
model = MultinomialNB()
model.fit(X_train, y_train)

#Model Evaluation
# Predict the labels on validation dataset

y_pred = model.predict(X_test)

# Print the accuracy
print(f'Accuracy: {accuracy_score(y_test, y_pred)}')
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

[ ]:
```
#Try SVM and Random Forest

# Import the necessary libraries
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier

# Train the model
model = SVC()
model.fit(X_train, y_train)

# Predict the labels on validation dataset
y_pred = model.predict(X_test)

# Print the accuracy
print(f'Accuracy: {accuracy_score(y_test, y_pred)}')
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

[ ]:
```
#Use sum balanced
# Train the model
model = SVC(class_weight='balanced')

model.fit(X_train, y_train)

# Predict the labels on validation dataset
y_pred = model.predict(X_test)

# Print the accuracy
print(f'Accuracy: {accuracy_score(y_test, y_pred)}')
```

```
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

[ ]:
```
#Use random forest
# Train the model

model = RandomForestClassifier()
model.fit(X_train, y_train)

# Predict the labels on validation dataset
y_pred = model.predict(X_test)

# Print the accuracy
print(f'Accuracy: {accuracy_score(y_test, y_pred)}')
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

[44]:
```
#Try using neural network

# Import the necessary libraries
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Assume training_df is your training data and 'Cleaned_Phrase' is the text data
texts = training_df['Cleaned_Phrase'].values
labels = training_df['Sentiment'].values + 1  # This will convert -1 to 0, 0 to␣
 ↪1, and 1 to 2

# Tokenization and Padding
tokenizer = Tokenizer()
tokenizer.fit_on_texts(texts)
sequences = tokenizer.texts_to_sequences(texts)
data = pad_sequences(sequences)

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.
 ↪2, random_state=0)

# Building the Neural Network
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(input_dim=len(tokenizer.word_index) + 1,␣
 ↪output_dim=128, input_length=data.shape[1]),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(3, activation='softmax')  # Assuming 3 sentiment␣
 ↪classes: 0, 1, 2 now instead of -1, 0, 1
])
```

```
# Compiling the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',␣
 ↪metrics=['accuracy'])

# Training the model
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=5,␣
 ↪batch_size=32)

# Evaluating the model
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Accuracy: {accuracy}')
```

```
Epoch 1/5
15/15 [==============================] - 0s 7ms/step - loss: 1.0648 - accuracy:
0.4630 - val_loss: 0.9759 - val_accuracy: 0.5776
Epoch 2/5
15/15 [==============================] - 0s 2ms/step - loss: 0.9961 - accuracy:
0.4783 - val_loss: 0.9979 - val_accuracy: 0.5776
Epoch 3/5
15/15 [==============================] - 0s 2ms/step - loss: 0.9196 - accuracy:
0.5739 - val_loss: 1.0058 - val_accuracy: 0.5776
Epoch 4/5
15/15 [==============================] - 0s 2ms/step - loss: 0.8159 - accuracy:
0.7043 - val_loss: 0.9928 - val_accuracy: 0.5776
Epoch 5/5
15/15 [==============================] - 0s 2ms/step - loss: 0.6774 - accuracy:
0.8304 - val_loss: 0.9990 - val_accuracy: 0.5776
4/4 [==============================] - 0s 909us/step - loss: 0.9990 - accuracy:
0.5776
Accuracy: 0.5775862336158752
```