[ ]:

```
#train count vectoriser using vocabulary
vectorizer = CountVectorizer(analyzer='char',
                             ngram_range=(3, 3),
                            vocabulary=vocab)

#create feature matrix for training set
corpus = train['text']
X = vectorizer.fit_transform(corpus)
feature_names = vectorizer.get_feature_names()

train_feat = pd.DataFrame(data=X.toarray(),columns=feature_names)
```

[ ]:

```
#Scale feature matrix
train_min = train_feat.min()
train_max = train_feat.max()
train_feat = (train_feat - train_min)/(train_max-train_min)

#Add target variable
train_feat['lang'] = list(train['lang'])
```

[ ]:

```
#create feature matrix for validation set
corpus = valid['text']
X = vectorizer.fit_transform(corpus)

valid_feat = pd.DataFrame(data=X.toarray(),columns=feature_names)
valid_feat = (valid_feat - train_min)/(train_max-train_min)
valid_feat['lang'] = list(valid['lang'])

#create feature matrix for test set
corpus = test['text']
X = vectorizer.fit_transform(corpus)

test_feat = pd.DataFrame(data=X.toarray(),columns=feature_names)
test_feat = (test_feat - train_min)/(train_max-train_min)
test_feat['lang'] = list(test['lang'])
```

[ ]:
```
train_feat.to_csv('./Feature_Files/train.csv')
valid_feat.to_csv('./Feature_Files/valid.csv')
test_feat.to_csv('./Feature_Files/test.csv')
```

[ ]:
```
train_feat = pd.read_csv("./Feature_Files/train.csv",index_col =0)
valid_feat = pd.read_csv("./Feature_Files/valid.csv",index_col =0)
test_feat = pd.read_csv("./Feature_Files/test.csv",index_col =0)
print(len(train_feat),len(valid_feat),len(test_feat))
train_feat.head()
```

```
280000 80000 40000
```

[ ]:
```
   e d            une      n w  t a     in  …   ica     t i  rea   \
0  0.4  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  …  0.0  0.0  0.0  0.0
1  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  …  0.0  0.0  0.0  0.0
2  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  …  0.0  0.0  0.0  0.0
3  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  …  0.0  0.0  0.0  0.0
4  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  …  0.0  0.0  0.0  0.0

         nn       per  lang
0  0.0  0.0  0.0  0.0  0.0   deu
1  0.0  0.0  0.0  0.0  0.0   por
2  0.0  0.0  0.0  0.0  0.0   cmn
3  0.0  0.0  0.0  0.0  0.0   mar
4  0.0  0.0  0.0  0.0  0.0   por

[5 rows x 1227 columns]
```

[ ]:
```
len(train_feat.columns)
```

[ ]:
```
1227
```

[ ]:
```
from sklearn.preprocessing import LabelEncoder
from keras.utils import np_utils

#Fit encoder
encoder = LabelEncoder()
encoder.fit(['deu', 'eng', 'fra', 'ita', 'por', 'cmn', 'jpn', 'mar'])

def encode(y):
    """
    Returns a list of one hot encodings

    Params
    ---------
        y: list of language labels
    """
    y_encoded = encoder.transform(y)
    y_dummy = np_utils.to_categorical(y_encoded)

    return y_dummy
```

```
2023-04-01 20:08:53.492195: I tensorflow/core/platform/cpu_feature_guard.cc:193]
This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical
operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
```