In [1]:
```
import nltk
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import os
os.chdir("D:/PlayGround/NLP/")
```

In [2]:
```
df_init = pd.read_excel('keyword_grouping.xlsx')
```

In [3]:
```
df_init.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6554 entries, 0 to 6553
Data columns (total 2 columns):
Keyword            6552 non-null object
Refined Keyword    3391 non-null object
dtypes: object(2)
memory usage: 102.5+ KB
```

In [5]:
```
df = df_init.copy()
```

In [6]:
```
# Remove NaN/Empty
df = df.dropna()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3391 entries, 77 to 6553
Data columns (total 2 columns):
Keyword            3391 non-null object
Refined Keyword    3391 non-null object
dtypes: object(2)
memory usage: 79.5+ KB
```

In [18]:
```
# function to add language feature

def add_lang_feat(dfi):
    dfi['non_eng'] = 0
    for index,row in dfi.iterrows():
        maxchar = max(row['Keyword'])
        if ord(maxchar)>127:
            dfi['non_eng'][index] = 1
    return dfi
```

In [19]:
```
# functions to split urls into words

def split_url(line):
    import re
    words = re.split('\:|\/|, |\.|\n|\>|\<|\-|\@',line)
    non_empty = []
    for word in words:
        if len(word)>0:
            non_empty.append(word)
    return ' '.join(non_empty)

def split_url_into_words(dfi):
    for index, row in dfi.iterrows():
        dfi['Keyword'][index] = split_url(str(row['Keyword']))
    return dfi
```

In [20]:
```
# function to detect language in the text and add a feature to df

def detect_lang(dfi):
    dfi['lang'] = 'en'
    from langdetect import detect
    for index, row in dfi.iterrows():
        try:
            dfi['lang'][index] = detect(row['Keyword'])
        except:
            print(row)
    return dfi
```

In [21]:
```
# separating df with only english characters
def eng_df(dfi):
    dfi = dfi[dfi['non_eng']==0]
    return dfi
```

In [12]:
```
# Global Count Feature Vector Object
from sklearn.feature_extraction.text import CountVectorizer
global_count_vect = CountVectorizer(analyzer='word')
```

In [22]:
```
# function to fit count feature vectors

def fit_count_vector(df_col):
    global_count_vect.fit(df_col)
```

In [23]:
```
# transform the data using count vectorizer object

def add_count_features(dfi):
    keyword_feature_matrix =  global_count_vect.transform(dfi['Keyword'])
    features = global_count_vect.get_feature_names()
    transformed_df = pd.DataFrame(keyword_feature_matrix.toarray(), columns = feat
    transformed_df['label'] = dfi['Refined Keyword'].tolist()
    transformed_df['Keyword'] = dfi['Keyword'].tolist()
    transformed_df['lang'] = dfi['lang'].tolist()
    return transformed_df
```

In [35]:
```
# function to convert categorial cols to integers with LabelEncoding

def label_encode(dfi, cat_cols):
    from sklearn import preprocessing
    for col in cat_cols:
        lbl = preprocessing.LabelEncoder()
        lbl.fit(list(dfi[col].values.astype('str')))
        dfi[col] = lbl.transform(list(dfi[col].values.astype('str')))
    return dfi
```

Preprocessing and Feature Engineering

In [17]:
```
# Step 1 : Remove NaN/Empty
df = df.dropna()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3391 entries, 77 to 6553
Data columns (total 2 columns):
Keyword            3391 non-null object
Refined Keyword    3391 non-null object
dtypes: object(2)
memory usage: 79.5+ KB
```

In [24]:
```
# Step 2 : Adding Language Feature
df = add_lang_feat(df)
df[:5]
```

Out[24]:

	Keyword	Refined Keyword	non_eng
77	1mobile market	Mobile App Development	0
78	2 2 channel ukraine	GL - Ukraine	0
79	5 ETAPAS DE BPMN	Technical Keyword	0
80	5 ETAPAS DEL BPMN	Technical Keyword	0
81	5 g live mobile	Mobile App Development	0

In [25]:
```
# Step 3 : Split urls in Keyword column to words
df = split_url_into_words(df)
```

In [26]:
```
# Step 4 : detect language in the text and add a feature to df
df = detect_lang(df)
```

In [27]:
```
df['lang'].value_counts()[:5]
```

Out[27]:
```
en    1650
ca     447
it     323
tl     220
es     122
Name: lang, dtype: int64
```

In [28]:
```
# Step 5 : separating df with only english characters
df = eng_df(df)
```

In [30]:
```
# Step 6 : fit count feature vectors
fit_count_vector(df['Keyword'])
```

In [31]:
```
# Step 7 : Add count feature vectors to df
df = add_count_features(df)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3310 entries, 0 to 3309
Columns: 2368 entries, 003 to lang
dtypes: int64(2365), object(3)
memory usage: 59.8+ MB
```

In [36]:
```
# Step 8 : Label encoding categorial columns
df = label_encode(df, ['lang'])
```

Training Logistic Regression Classifier Model using Cross Validation

In [32]:
```
# custom cross validation function

def custom_cv(classifier_obj, X, y, split=5):
    from sklearn.model_selection import cross_val_score
    scores = cross_val_score(classifier_obj, X, y, cv=split)
    print(scores)
    return scores.mean()
```

In [38]:
```
from sklearn.linear_model import LogisticRegression
x_col = list(set(list(df)) - set(['label', 'Keyword']))
y_col = ['label']
accuracy_lc = custom_cv(LogisticRegression(), df[x_col], df[y_col], split=10)
print(accuracy_lc)
```

```
[0.64305949 0.69942197 0.75892857 0.81626506 0.8        0.77575758
 0.87116564 0.89814815 0.81875    0.76038339]
0.7841879841731413
```

Training Random Forest Classifier Model using Cross Validation

In [39]:
```
from sklearn.ensemble import RandomForestClassifier
accuracy_rf = custom_cv(RandomForestClassifier(), df[x_col], df[y_col], split=10)
print(accuracy_rf)
```

```
[0.57790368 0.63294798 0.68452381 0.72289157 0.75757576 0.73636364
 0.77300613 0.82407407 0.740625   0.66134185]
0.7111253491404965
```

Seperating Training and Demo data for Prediction

In [42]:
```
df_demo = df_init.dropna()
df_demo.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3391 entries, 77 to 6553
Data columns (total 2 columns):
Keyword            3391 non-null object
Refined Keyword    3391 non-null object
dtypes: object(2)
memory usage: 79.5+ KB
```

In [43]:
```
import numpy as np
indices = np.random.rand(len(df_demo)) <= 0.05
df_demo = df_demo[indices]
df_demo.to_csv('demo.csv', index=False)
```

In [45]:
```
training_df = df_init.dropna()
training_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3391 entries, 77 to 6553
Data columns (total 2 columns):
Keyword            3391 non-null object
Refined Keyword    3391 non-null object
dtypes: object(2)
memory usage: 79.5+ KB
```

In [46]:
```
training_df = training_df[~indices]
training_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3218 entries, 77 to 6553
Data columns (total 2 columns):
Keyword            3218 non-null object
Refined Keyword    3218 non-null object
dtypes: object(2)
memory usage: 75.4+ KB
```

In [47]:
```
training_df.to_csv('training_data.csv', index=False)
```

Preprocessing function for Prediction Flow

In [44]:
```
def preprocess_pred(dfi):

    # Step 1 : Remove NaN/Empty
    dfi = dfi.dropna()

    # Step 2 : Adding Language Feature
    dfi = add_lang_feat(dfi)

    # Step 3 : Split urls in Keyword column to words
    dfi = split_url_into_words(dfi)

    # Step 4 : detect language in the text and add a feature to df
    dfi = detect_lang(dfi)

    # Step 5 : separating df with only english characters
    dfi = eng_df(dfi)

    # Step 6 : Add count feature vectors to df
    dfi = add_count_features(dfi)

    # Step 8 : Label encoding categorial columns
    dfi = label_encode(dfi, ['lang'])

    return dfi
```

In [48]:
```
# Loading training data
train_df = pd.read_csv('training_data.csv')
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3218 entries, 0 to 3217
Data columns (total 2 columns):
Keyword            3218 non-null object
Refined Keyword    3218 non-null object
dtypes: object(2)
memory usage: 50.4+ KB
```

In [49]:
```
# preprocessing training data
pp_train_df = preprocess_pred(train_df)
pp_train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3142 entries, 0 to 3141
Columns: 2368 entries, 003 to lang
dtypes: int64(2366), object(2)
memory usage: 56.8+ MB
```

In [50]:
```
# loading demo data
test_df = pd.read_csv('demo.csv')
test_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 173 entries, 0 to 172
Data columns (total 2 columns):
Keyword            173 non-null object
Refined Keyword    173 non-null object
dtypes: object(2)
memory usage: 2.8+ KB
```

In [51]:
```
# preprocessing demo data
pp_test_df = preprocess_pred(test_df)
pp_test_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 168 entries, 0 to 167
Columns: 2368 entries, 003 to lang
dtypes: int64(2366), object(2)
memory usage: 3.0+ MB
```

Training Logistic Regression Model and Predicting on Demo Data

In [52]:
```
from sklearn.linear_model import LogisticRegression
x_cols = list(set(list(pp_train_df)) - set(['label', 'Keyword']))
y_cols = ['label']
lr_model = LogisticRegression().fit(pp_train_df[x_cols],pp_train_df[y_cols] )
predictions = lr_model.predict(pp_test_df[x_cols])
```

In [53]:
```
results_df = pp_test_df[['Keyword','label']]
results_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 168 entries, 0 to 167
Data columns (total 2 columns):
Keyword    168 non-null object
label      168 non-null object
dtypes: object(2)
memory usage: 2.7+ KB
```

In [54]:
```
results_df['predicted'] = predictions
```

In [60]:
```
results_df[:5]
```

Out[60]:

	Keyword	label	predicted
0	50358906549 Shashikant Chaudhary	GlobalLogic	Careers
1	net trainee global logic	Careers	Careers
2	"Global Logic Israel"	GL - Israel	GL - Israel
3	About globallogic technology	GlobalLogic	GlobalLogic
4	address of global logic technologies limited	GlobalLogic	GlobalLogic

In [58]:
```
results_df.to_csv('results.csv', index=False)
```