Language detection with stopword and tokenization in NLTK

There are several ways to do text language classification. The esiest way is a stopwords based approach. The term “stopword” is used in natural language processing to refer words which should be filtered out from text before doing any kind of processing.

In [1]:
```
#these are the dependencies related to NLP that we need to import before proce
eding
import nltk
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk import wordpunct_tokenize
from nltk.corpus import stopwords
```

In [2]:
```
#any text in any language that you want to know the Language of.

abc = "Türk Dil Kurumu, Türkiye Türkçesinin imlasının standartlarını belirlem
e, dil hakkında çeşitli düzeyde çalışmalar yapma gibi konularda faaliyetler yü
rütür. Günümüzde bu kurum, sadece Türkiye değil, dünya çapında Türkçe ve Türko
loji ile ilgili çeşitli çalışmalarda kurumsal yüzü veya akademik mensuplarıyla
 yer almaktadır."
```

so we have a text whose language we want to detect depending on stopwords being used in such text. First step is to “tokenize” - convert given text to a list of “words” or “tokens” - using an approach or another depending on our requeriments.

In this case we are going to split all punctuations into separate tokens

In [3]:
```
#tokenizing the above text along with punctuation it cantain and print it.

print(wordpunct_tokenize(abc))
```

```
['Türk', 'Dil', 'Kurumu', ',', 'Türkiye', 'Türkçesinin', 'imlasının', 'standa
rtlarını', 'belirleme', ',', 'dil', 'hakkında', 'çeşitli', 'düzeyde', 'çalışm
alar', 'yapma', 'gibi', 'konularda', 'faaliyetler', 'yürütür', '.', 'Günümüzd
e', 'bu', 'kurum', ',', 'sadece', 'Türkiye', 'değil', ',', 'dünya', 'çapınd
a', 'Türkçe', 've', 'Türkoloji', 'ile', 'ilgili', 'çeşitli', 'çalışmalarda',
 'kurumsal', 'yüzü', 'veya', 'akademik', 'mensuplarıyla', 'yer', 'almaktadı
r', '.']
```

In [4]:
```
#these are the languages from where you can use stopwords directly. if your la
nuage is not here then you can append stopwords
#simply by append function

stopwords.fileids()
```

Out[4]:
```
['danish',
 'dutch',
 'english',
 'finnish',
 'french',
 'german',
 'hungarian',
 'italian',
 'kazakh',
 'norwegian',
 'portuguese',
 'russian',
 'spanish',
 'swedish',
 'turkish']
```

In [5]:
```
#here we can check the stopwords nltk have for perticular language

stopwords.words('turkish') [0:10]
```

Out[5]:
```
['acaba',
 'ama',
 'aslında',
 'az',
 'bazı',
 'belki',
 'biri',
 'birkaç',
 'birşey',
 'biz']
```

In [6]:
```
languages_ratios = {}
```

In [7]:
```
tokens = wordpunct_tokenize(abc)
words = [word.lower() for word in tokens]
```

In [8]:
```
#Now we need to compute language probability depending on which stopwords are
 used

for language in stopwords.fileids():
    stopwords_set = set(stopwords.words(language))
    words_set = set()
    common_elements = words_set.intersection(stopwords_set)
    languages_ratios[abc] = len(common_elements)
```

In [9]:
```
languages_ratios
```

Out[9]:
```
{'Türk Dil Kurumu, Türkiye Türkçesinin imlasının standartlarını belirleme, di
l hakkında çeşitli düzeyde çalışmalar yapma gibi konularda faaliyetler yürütü
r. Günümüzde bu kurum, sadece Türkiye değil, dünya çapında Türkçe ve Türkoloj
i ile ilgili çeşitli çalışmalarda kurumsal yüzü veya akademik mensuplarıyla y
er almaktadır.': 0}
```

First we tokenize using wordpunct_tokenize function and lowercase all splitted tokens, then we walk across nltk included languages and count how many unique stopwords are seen in analyzed text to put this in “language_ratios” dictionary.

In [10]:
```
most_rated_language = max(languages_ratios, key=languages_ratios.get)
most_rated_language
```

Out[10]:
```
'Türk Dil Kurumu, Türkiye Türkçesinin imlasının standartlarını belirleme, dil
hakkında çeşitli düzeyde çalışmalar yapma gibi konularda faaliyetler yürütür.
Günümüzde bu kurum, sadece Türkiye değil, dünya çapında Türkçe ve Türkoloji i
le ilgili çeşitli çalışmalarda kurumsal yüzü veya akademik mensuplarıyla yer
 almaktadır.'
```

In [11]:
```
print (language)
```

```
turkish
```

mission successful

it seems this approach works fine with well written texts and especially for those who follow grammatical rules and it's really easy to implement.