Text Normalization

It involves cleaning and preprocessing text data to make it consistent and usable for different NLP tasks. The process includes a variety of techniques, such as case normalization, punctuation removal, stop word removal, stemming, and lemmatization.

Import Library (nltk)

- work with human language data for applying in statistical natural language processing (NLP).
- It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning
- Ref : "https://www.nltk.org/data.html"
- NLTK **Corpora (DataSets)** : "https://www.nltk.org/nltk_data/"

In [68]:
```
import nltk
nltk.download() # to download all features
```
```
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml
```
Out[68]:
```
True
```

In [39]:
```
# to know the functions in the library
dir(nltk)[0:10]
```
Out[39]:
```
['ARLSTem',
 'ARLSTem2',
 'AbstractLazySequence',
 'AffixTagger',
 'AlignedSent',
 'Alignment',
 'AnnotationTask',
 'ApplicationExpression',
 'Assignment',
 'BigramAssocMeasures']
```

Tokenization

Tokenization : mean split the sentence into tokens(**words**) and called **Word tokenizer**

Tokenization : split paragraph into sentence and called **sentence tokenizer**

- Hint Word tokenizer : is like split but split depend on space or symbol to seprate words
- Apply tokenization Text -> **This product is amazing, but the delivery was late.**
  - **Answer** : ["This", "product", "is", "amazing", ",", "but", "the", "delivery", "was", "late", "."]

Tokenization Using NLTK EN

In [69]:
```
from nltk.tokenize import word_tokenize
#word_tokenize is a function
EXAMPLE_TEXT = """
Hello Mr. Smith, how are you doing today? The weather is great,
and Python is awesome. The sky is pinkish-blue. You shouldn't eat cardboard.
"""
print(word_tokenize(EXAMPLE_TEXT))
```
```
['Hello', 'Mr.', 'Smith', ',', 'how', 'are', 'you', 'doing', 'today', '?', 'The', 'weather', 'is', 'great', ',', 'and', 'Python', 'is', 'awesome', '.', 'The', 'sky', 'is', 'pinkish-blue', '.', 'You', 'should', "n't", 'eat', 'cardboar
d', '.']
```

Difference between split and token

In [70]:
```
text = "my name's hossam and i work as a teacher assistant at bfcai and i have 600$"
print(text.split())
print('------------------------')
print(word_tokenize(text))
print('====================================================')
```
```
['my', "name's", 'hossam', 'and', 'i', 'work', 'as', 'a', 'teacher', 'assistant', 'at', 'bfcai', 'and', 'i', 'have', '600$']
------------------------
['my', 'name', "'s", 'hossam', 'and', 'i', 'work', 'as', 'a', 'teacher', 'assistant', 'at', 'bfcai', 'and', 'i', 'have', '600', '$']
====================================================
```

In [71]:
```
from nltk.tokenize import sent_tokenize
s = 'Good muffins cost $3.88\nin New York.  Please buy me two of them.\n\nThanks.'
print(sent_tokenize(s))
#whitespaceTokenizer?
```
```
['Good muffins cost $3.88\nin New York.', 'Please buy me two of them.', 'Thanks.']
```

Tokenization Using NLTK AR

In [72]:
```
EXAMPLE_TEXT='هل تعلم؟ #نيوكاسل يتفوق بالمواجهات المباشرة على #ارسنال في تاريخ الدوري الممتاز الانجليزي؟ بعد قليل أرقام واحصائيات للفريقين'
print(word_tokenize(EXAMPLE_TEXT))
```
```
['هل', 'تعلم؟', '#', 'نيوكاسل', 'يتفوق', 'بالمواجهات', 'المباشرة', 'على', '#', 'ارسنال', 'في', 'تاريخ', 'الدوري', 'الممتاز', 'الانجليزي؟', 'بعد', 'قليل', 'أرقام', 'واحصائيات', 'للفريقين']
```

In [74]:
```
print(EXAMPLE_TEXT.split())
print('------------------------')
print(word_tokenize(EXAMPLE_TEXT))
print('====================================================')
```
```
['هل', 'تعلم؟', '#نيوكاسل', 'يتفوق', 'بالمواجهات', 'المباشرة', 'على', '#ارسنال', 'في', 'تاريخ', 'الدوري', 'الممتاز', 'الانجليزي؟', 'بعد', 'قليل', 'أرقام', 'واحصائيات', 'للفريقين']
------------------------
['هل', 'تعلم؟', '#', 'نيوكاسل', 'يتفوق', 'بالمواجهات', 'المباشرة', 'على', '#', 'ارسنال', 'في', 'تاريخ', 'الدوري', 'الممتاز', 'الانجليزي؟', 'بعد', 'قليل', 'أرقام', 'واحصائيات', 'للفريقين']
====================================================
```

Text Normalizaion (Cleaning)

How to remove all this from the text?

- **Stop Words** : (such as "the", "a", "an", "in") -- a search engine has been programmed to ignore.
- **Punctuation** : ['!', '"', '#', '$', '%', '&', "'", '(', ')', '*', '+', ',....]
- **Stemming** : Stemming reduce the text by a set of pre-defined rules like removing `ing` from verbs
  - Stemming is the process of producing morphological variants of a root/base word.
  - اداه تسمح بتجريد اى كلمه من جميع الاضافات التى فيها والعوده للمصدر الاصلى لها
  - Example : (plays,played,playing,player) -->> play
- **Lemmatization** : reduce the word by looking it up in the `WordNet` where it tries to find the root of the word for example `rocks` -> `rock`
  - Lemmatization is similar to stemming but it brings context to the words.
  - So it links words with similar meanings to one word.
  - اكثر قوة وفعاليه لانها مش بتكتفى انها تحذف الزوائد من الكلمات ولكن بتبحث ف معنى واصل الكلمه
  - Example : (rocks - rock) , (corpora - corpus) , (better - good)

Lemmatization vs Stemming

- The key concept here is that stemming sometime destroy the word unlike lemmatization where we keep the meaning.

stop words in EN

In [41]:
```
#NLTK(Natural Language Toolkit) in python has a list of stopwords stored in 16 different languages
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
```

In [42]:
```
print(stopwords.words('english'))
```
```
['a', 'about', 'above', 'after', 'again', 'against', 'ain', 'all', 'am', 'an', 'and', 'any', 'are', 'aren', "aren't", 'as', 'at', 'be', 'because', 'been', 'before', 'being', 'below', 'between', 'both', 'but', 'by', 'can', 'couldn', "co
uldn't", 'd', 'did', 'didn', "didn't", 'do', 'does', 'doesn', "doesn't", 'doing', 'don', "don't", 'down', 'during', 'each', 'few', 'for', 'from', 'further', 'had', 'hadn', "hadn't", 'has', 'hasn', "hasn't", 'have', 'haven', "haven't",
'having', 'he', "he'd", "he'll", 'her', 'here', 'hers', 'herself', "he's", 'him', 'himself', 'his', 'how', 'i', "i'd", 'if', "i'll", "i'm", 'in', 'into', 'is', 'isn', "isn't", 'it', "it'd", "it'll", "it's", 'its', 'itself', "i've", 'ju
st', 'll', 'm', 'ma', 'me', 'mightn', "mightn't", 'more', 'most', 'mustn', "mustn't", 'my', 'myself', 'needn', "needn't", 'no', 'nor', 'not', 'now', 'o', 'of', 'off', 'on', 'once', 'only', 'or', 'other', 'our', 'ours', 'ourselves', 'ou
t', 'over', 'own', 're', 's', 'same', 'shan', "shan't", 'she', "she'd", "she'll", "she's", 'should', 'shouldn', "shouldn't", "should've", 'so', 'some', 'such', 't', 'than', 'that', "that'll", 'the', 'their', 'theirs', 'them', 'themselv
es', 'then', 'there', 'these', 'they', "they'd", "they'll", "they're", "they've", 'this', 'those', 'through', 'to', 'too', 'under', 'until', 'up', 've', 'very', 'was', 'wasn', "wasn't", 'we', "we'd", "we'll", "we're", 'were', 'weren',
"weren't", "we've", 'what', 'when', 'where', 'which', 'while', 'who', 'whom', 'why', 'will', 'with', 'won', "won't", 'wouldn', "wouldn't", 'y', 'you', "you'd", "you'll", 'your', "you're", 'yours', 'yourself', 'yourselves', "you've"]
```

Task 1 : Removing stop words with NLTK ?

In [ ]:

In [ ]:

In [ ]:

In [15]:
```
example_sent = """This is a sample sentence, showing off the stop words filtration."""
stop_words = stopwords.words('english')
word_tokens = word_tokenize(example_sent)
filtered_sentence = []
for w in word_tokens:
    if w.lower() not in stop_words:
        filtered_sentence.append(w.lower())
```
```
['This', 'is', 'a', 'sample', 'sentence', ',', 'showing', 'off', 'the', 'stop', 'words', 'filtration', '.']
['sample', 'sentence', ',', 'showing', 'stop', 'words', 'filtration', '.']
sample sentence , showing stop words filtration .
```

stop words in AR

In [88]:
```
stop_word_arabic = set(stopwords.words("arabic"))
print(len(stop_word_arabic))
```
```
701
```

In [90]:
```
' - '.join(stop_word_arabic)
```
Out[90]:
```
[illegible]
```

punctuation

In [17]:
```
from string import punctuation

punctuation = list(punctuation)
print(punctuation)
```
```
['!', '"', '#', '$', '%', '&', "'", '(', ')', '*', '+', ',', '-', '.', '/', ':', ';', '<', '=', '>', '?', '@', '[', '\\', ']', '^', '_', '`', '{', '|', '}', '~']
```

Task 2 : Removing punctuation with tokenization ?

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [18]:
```
example_sent = """My Email address is: taneshbalodi8@gmail.com."""
word_tokens = word_tokenize(example_sent)
filtered_sentence = []

for w in word_tokens:
    if w not in punctuation:
        filtered_sentence.append(w)

print(word_tokens)
print(filtered_sentence)
```
```
['My', 'Email', 'address', 'is', ':', 'taneshbalodi8', '@', 'gmail.com', '.']
['My', 'Email', 'address', 'is', 'taneshbalodi8', 'gmail.com']
```

Task 3 : Removing punctuation without tokenization

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [19]:
```
example_sent = """My Email address is: hossamfares180100@gmail.com."""
filtered_sentence = []

for w in example_sent:
    if w not in punctuation:
        filtered_sentence.append(w)

print(example_sent)
print(filtered_sentence)
print(''.join(filtered_sentence))
```
```
My Email address is: hossamfares180100@gmail.com.
['M', 'y', ' ', 'E', 'm', 'a', 'i', 'l', ' ', 'a', 'd', 'd', 'r', 'e', 's', 's', ' ', 'i', 's', ' ', 'h', 'o', 's', 's', 'a', 'm', 'f', 'a', 'r', 'e', 's', '1', '8', '0', '1', '0', '0', 'g', 'm', 'a', 'i', 'l', 'c', 'o', 'm']
My Email address is hossamfares180100gmailcom
```

In [ ]:

stemming using nltk

In [21]:
```
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize

ps = PorterStemmer()

# choose some words to be stemmed
words = ["program", "programs", "programmer", "programming", "programmers"]

for w in words:
    print(w, " : ", ps.stem(w))
```
```
program  :  program
programs  :  program
programmer  :  programm
programming  :  program
programmers  :  programm
```

In [22]:
```
sentence = "Programmers program with programming languages"
words = word_tokenize(sentence)


for w in words:
    print(w, " : ", ps.stem(w))
```
```
Programmers  :  programm
program  :  program
with  :  with
programming  :  program
languages  :  languag
```

HINT: snowballstemmer is somewhat faster and more logical than the original Porter Stemmer.

In [23]:
```
from nltk.stem import SnowballStemmer
snowball = SnowballStemmer(language='english')
words = ['generous','generate','generously','generation']
for word in words:
    print(word,"--->",snowball.stem(word))
```
```
generous ---> generous
generate ---> generat
generously ---> generous
generation ---> generat
```

Snowball stemmer support different languages

In [40]:
```
from nltk.stem.snowball import SnowballStemmer

print(", ".join(SnowballStemmer.languages))
```
```
arabic, danish, dutch, english, finnish, french, german, hungarian, italian, norwegian, porter, portuguese, romanian, russian, spanish, swedish
```

stemming in AR

In [24]:
```
#stemming is very weak in arabic
words = ['الجري','تجري','يجرون','جري','يجري']
for word in words:
    print(word+' --> '+ ps.stem(word))
```
```
الجري --> الجري
تجري --> تجري
يجرون --> يجرون
جري --> جري
يجري --> يجري
```

In [25]:
```
from nltk.stem.snowball import SnowballStemmer
s_stemmer = SnowballStemmer(language='arabic')

words = ['الجري','تجري','يجرون','جري','يجري']
for word in words:
    print(word+' --> '+s_stemmer.stem(word))
```
```
الجري --> الجر
تجري --> تجر
يجرون --> يجرو
جري --> جر
يجري --> يجر
```

In [26]:
```
from nltk.stem.isri import ISRIStemmer
st = ISRIStemmer()
words = ['الجري','تجري','يجرون','جري','يجري']

for word in words :
    print(st.stem(word))
print (st.stem('العلميون'))
```
```
جري
تجر
يجر
جري
يجر
علم
```

Lemmatization using nltk

- Examples of lemmatization:
  - rocks : rock
  - corpora : corpus
  - better : good

In [92]:
```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()
```

In [94]:
```
print("rocks :", lemmatizer.lemmatize("rocks"))
print("corpora :", lemmatizer.lemmatize("corpora"))
print("better :", lemmatizer.lemmatize("better", pos ="a"))
```
```
rocks : rock
corpora : corpus
better : good
```

In [96]:
```
from nltk.stem import WordNetLemmatizer

# Without POS (default is noun)
print(lemmatizer.lemmatize("better"))
# With POS as adjective ('a')
print(lemmatizer.lemmatize("better", pos="a"))
# With POS as verb ('v')
print(lemmatizer.lemmatize("running", pos="v"))
```
```
better
good
run
```

In [98]:
```
# single word lemmatization examples
list1 = ['kites', 'babies', 'dogs', 'flying', 'smiling', 'driving', 'died', 'tried', 'feet']

for words in list1:
    print(words + " ---> " + lemmatizer.lemmatize(words))
```
```
kites ---> kite
babies ---> baby
dogs ---> dog
flying ---> flying
smiling ---> smiling
driving ---> driving
died ---> died
tried ---> tried
feet ---> foot
```

In [30]:
```
words = ["is","was","be","been","are","were"]

for word in words :
    print(lemmatizer.lemmatize(word))
```
```
is
wa
be
been
are
were
```

In [31]:
```
#what if we put them as verb
words = ["is","was","be","been","are","were"]
for word in words :
    print(lemmatizer.lemmatize(word,'v'))
```
```
be
be
be
be
be
be
```

lemmatization in AR

In [32]:
```
words = ['الجري','تجري','يجرون','جري','يجري']

for word in words :
    print(lemmatizer.lemmatize(word))
```
```
الجري
تجري
يجرون
جري
يجري
```

Task 4 : Create Method Take (Text as input) then : (15 MIN)

- returns a cleaned version by removing stopwords, punctuation, applying lemmatization, and stemming:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [114…
```
def clean_text(text):
    filtered_sentence = []

    for word in text.split(" "):
        if word.lower() not in stop_words and  w not in punctuation :
            filtered_sentence.append(word.lower())
    #print(filtered_sentence)

    ps_text = [ps.stem(word) for word in filtered_sentence]
    lemmatizer_text = [lemmatizer.lemmatize(word) for word in filtered_sentence]
    return ps_text , lemmatizer_text
```

In [118…
```
clean_text("Welcome to the NLP Course!. Natural Language Processing (NLP) is an exciting field of Artificial Intelligence that enables computers to understand, interpret, and generate human language.")
```
Out[118…
```
(['welcom',
  'nlp',
  'course!.',
  'natur',
  'languag',
  'process',
  '(nlp)',
  'excit',
  'field',
  'artifici',
  'intellig',
  'enabl',
  'comput',
  'understand,',
  'interpret,',
  'gener',
  'human',
  'language.'],
 ['welcome',
  'nlp',
  'course!.',
  'natural',
  'language',
  'processing',
  '(nlp)',
  'exciting',
  'field',
  'artificial',
  'intelligence',
  'enables',
  'computer',
  'understand,',
  'interpret,',
  'generate',
  'human',
  'language.'])
```

In [ ]:

In [ ]:

In [ ]:

Extra Topics :

Some works in Arabic

In [33]:
```
pip install qalsadi
```
```
Collecting qalsadi
  Downloading qalsadi-0.5-py3-none-any.whl.metadata (12 kB)
Collecting Arabic-Stopwords>=0.4.2 (from qalsadi)
  Downloading Arabic_Stopwords-0.4.3-py3-none-any.whl.metadata (8.9 kB)
Collecting alyahmor>=0.2 (from qalsadi)
  Downloading alyahmor-0.2-py3-none-any.whl.metadata (11 kB)
Collecting arramooz-pysqlite>=0.4.2 (from qalsadi)
  Downloading arramooz_pysqlite-0.4.2-py3-none-any.whl.metadata (4.0 kB)
Collecting codernitydb3 (from qalsadi)
  Downloading codernitydb3-0.6.0.tar.gz (46 kB)
     ---------------------------------------- 0.0/46.1 kB ? eta -:--:--
     -------------------------- ------------- 30.7/46.1 kB 1.3 MB/s eta 0:00:01
     ---------------------------------------- 46.1/46.1 kB 568.6 kB/s eta 0:00:00
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Collecting libqutrub>=1.2.3 (from qalsadi)
  Downloading libqutrub-1.2.4.1-py3-none-any.whl.metadata (7.5 kB)
Collecting mysam-tagmanager>=0.3.3 (from qalsadi)
  Downloading mysam_tagmanager-0.4-py3-none-any.whl.metadata (10 kB)
Collecting naftawayh>=0.3 (from qalsadi)
  Downloading Naftawayh-0.4-py3-none-any.whl.metadata (8.7 kB)
Collecting pickledb>=0.9.2 (from qalsadi)
  Downloading pickledb-1.3.2.tar.gz (10 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Collecting pyarabic>=0.6.7 (from qalsadi)
  Downloading PyArabic-0.6.15-py3-none-any.whl.metadata (10 kB)
Collecting tashaphyne>=0.3.4.1 (from qalsadi)
  Downloading Tashaphyne-0.3.6-py3-none-any.whl.metadata (18 kB)
Requirement already satisfied: orjson in c:\users\fady\anaconda3\lib\site-packages (from pickledb>=0.9.2->qalsadi) (3.10.5)
Collecting aiofiles (from pickledb>=0.9.2->qalsadi)
  Downloading aiofiles-24.1.0-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: six>=1.14.0 in c:\users\fady\anaconda3\lib\site-packages (from pyarabic>=0.6.7->qalsadi) (1.16.0)
Downloading qalsadi-0.5-py3-none-any.whl (264 kB)
   ---------------------------------------- 0.0/264.3 kB ? eta -:--:--
   ------------ --------------------------- 81.9/264.3 kB 2.2 MB/s eta 0:00:01
   -------------------------- ------------- 174.1/264.3 kB 2.1 MB/s eta 0:00:01
   -------------------------------------- - 256.0/264.3 kB 2.0 MB/s eta 0:00:01
   ---------------------------------------- 264.3/264.3 kB 1.8 MB/s eta 0:00:00
Downloading alyahmor-0.2-py3-none-any.whl (65 kB)
   ---------------------------------------- 0.0/65.1 kB ? eta -:--:--
   ---------------------------------------- 65.1/65.1 kB 1.2 MB/s eta 0:00:00
Downloading Arabic_Stopwords-0.4.3-py3-none-any.whl (360 kB)
   ---------------------------------------- 0.0/360.5 kB ? eta -:--:--
   ------------ --------------------------- 112.6/360.5 kB 2.2 MB/s eta 0:00:01
   ---------------------- ----------------- 204.8/360.5 kB 2.1 MB/s eta 0:00:01
   ---------------------------------- ----- 307.2/360.5 kB 2.1 MB/s eta 0:00:01
   ---------------------------------------- 360.5/360.5 kB 2.0 MB/s eta 0:00:00
Downloading arramooz_pysqlite-0.4.2-py3-none-any.whl (6.3 MB)
   ---------------------------------------- 0.0/6.3 MB ? eta -:--:--
    --------------------------------------- 0.1/6.3 MB 2.3 MB/s eta 0:00:03
   - -------------------------------------- 0.2/6.3 MB 2.1 MB/s eta 0:00:03
   - -------------------------------------- 0.3/6.3 MB 2.2 MB/s eta 0:00:03
   -- ------------------------------------- 0.4/6.3 MB 2.2 MB/s eta 0:00:03
   --- ------------------------------------ 0.5/6.3 MB 2.4 MB/s eta 0:00:03
   --- ------------------------------------ 0.6/6.3 MB 2.2 MB/s eta 0:00:03
   ---- ----------------------------------- 0.7/6.3 MB 2.3 MB/s eta 0:00:03
   ----- ---------------------------------- 0.8/6.3 MB 2.3 MB/s eta 0:00:03
   ----- ---------------------------------- 0.9/6.3 MB 2.3 MB/s eta 0:00:03
   ------ --------------------------------- 1.0/6.3 MB 2.3 MB/s eta 0:00:03
   ------- -------------------------------- 1.1/6.3 MB 2.3 MB/s eta 0:00:03
   ------- -------------------------------- 1.2/6.3 MB 2.3 MB/s eta 0:00:03
   -------- ------------------------------- 1.3/6.3 MB 2.3 MB/s eta 0:00:03
   --------- ------------------------------ 1.4/6.3 MB 2.2 MB/s eta 0:00:03
   --------- ------------------------------ 1.5/6.3 MB 2.3 MB/s eta 0:00:03
   ---------- ----------------------------- 1.6/6.3 MB 2.3 MB/s eta 0:00:03
   ----------- ---------------------------- 1.8/6.3 MB 2.3 MB/s eta 0:00:03
   ----------- ---------------------------- 1.9/6.3 MB 2.3 MB/s eta 0:00:02
   ------------ --------------------------- 2.0/6.3 MB 2.3 MB/s eta 0:00:02
   ------------- -------------------------- 2.1/6.3 MB 2.3 MB/s eta 0:00:02
   ------------- -------------------------- 2.2/6.3 MB 2.3 MB/s eta 0:00:02
   -------------- ------------------------- 2.3/6.3 MB 2.3 MB/s eta 0:00:02
   --------------- ------------------------ 2.4/6.3 MB 2.3 MB/s eta 0:00:02
   --------------- ------------------------ 2.5/6.3 MB 2.3 MB/s eta 0:00:02
   ---------------- ----------------------- 2.6/6.3 MB 2.3 MB/s eta 0:00:02
   ---------------- ----------------------- 2.7/6.3 MB 2.3 MB/s eta 0:00:02
   ----------------- ---------------------- 2.8/6.3 MB 2.3 MB/s eta 0:00:02
   ------------------ --------------------- 2.9/6.3 MB 2.3 MB/s eta 0:00:02
   ------------------- -------------------- 3.0/6.3 MB 2.3 MB/s eta 0:00:02
   ------------------- -------------------- 3.1/6.3 MB 2.3 MB/s eta 0:00:02
   -------------------- ------------------- 3.2/6.3 MB 2.3 MB/s eta 0:00:02
   --------------------- ------------------ 3.3/6.3 MB 2.3 MB/s eta 0:00:02
   --------------------- ------------------ 3.4/6.3 MB 2.3 MB/s eta 0:00:02
   ---------------------- ----------------- 3.5/6.3 MB 2.3 MB/s eta 0:00:02
   ---------------------- ----------------- 3.6/6.3 MB 2.3 MB/s eta 0:00:02
   ----------------------- ---------------- 3.8/6.3 MB 2.3 MB/s eta 0:00:02
   ------------------------ --------------- 3.9/6.3 MB 2.3 MB/s eta 0:00:02
   ------------------------- -------------- 4.0/6.3 MB 2.3 MB/s eta 0:00:02
   ------------------------- -------------- 4.0/6.3 MB 2.3 MB/s eta 0:00:02
   ------------------------- -------------- 4.0/6.3 MB 2.3 MB/s eta 0:00:02
   ------------------------- -------------- 4.0/6.3 MB 2.3 MB/s eta 0:00:02
   ------------------------- -------------- 4.0/6.3 MB 2.3 MB/s eta 0:00:02
   ------------------------- -------------- 4.0/6.3 MB 2.3 MB/s eta 0:00:02
   ------------------------- -------------- 4.0/6.3 MB 2.3 MB/s eta 0:00:02
   ------------------------- -------------- 4.1/6.3 MB 2.0 MB/s eta 0:00:02
   ------------------------------ --------- 4.8/6.3 MB 2.3 MB/s eta 0:00:01
   ------------------------------ --------- 4.9/6.3 MB 2.3 MB/s eta 0:00:01
   ------------------------------- -------- 5.0/6.3 MB 2.3 MB/s eta 0:00:01
   -------------------------------- ------- 5.1/6.3 MB 2.3 MB/s eta 0:00:01
   -------------------------------- ------- 5.2/6.3 MB 2.3 MB/s eta 0:00:01
   --------------------------------- ------ 5.3/6.3 MB 2.3 MB/s eta 0:00:01
   ---------------------------------- ----- 5.4/6.3 MB 2.3 MB/s eta 0:00:01
   ----------------------------------- ---- 5.6/6.3 MB 2.3 MB/s eta 0:00:01
   ----------------------------------- ---- 5.6/6.3 MB 2.3 MB/s eta 0:00:01
   ------------------------------------ --- 5.8/6.3 MB 2.3 MB/s eta 0:00:01
   ------------------------------------- -- 5.9/6.3 MB 2.3 MB/s eta 0:00:01
   -------------------------------------- - 6.0/6.3 MB 2.3 MB/s eta 0:00:01
   -------------------------------------- - 6.1/6.3 MB 2.3 MB/s eta 0:00:01
   ---------------------------------------  6.2/6.3 MB 2.3 MB/s eta 0:00:01
   ---------------------------------------  6.3/6.3 MB 2.3 MB/s eta 0:00:01
   ---------------------------------------  6.3/6.3 MB 2.3 MB/s eta 0:00:01
   ---------------------------------------- 6.3/6.3 MB 2.2 MB/s eta 0:00:00
Downloading libqutrub-1.2.4.1-py3-none-any.whl (138 kB)
   ---------------------------------------- 0.0/139.0 kB ? eta -:--:--
   -------------------------- ------------- 92.2/139.0 kB 2.6 MB/s eta 0:00:01
   ---------------------------------------- 139.0/139.0 kB 2.0 MB/s eta 0:00:00
Downloading mysam_tagmanager-0.4-py3-none-any.whl (37 kB)
Downloading Naftawayh-0.4-py3-none-any.whl (332 kB)
   ---------------------------------------- 0.0/332.6 kB ? eta -:--:--
   ----------- ---------------------------- 92.2/332.6 kB 1.7 MB/s eta 0:00:01
   ------------------------ --------------- 204.8/332.6 kB 2.1 MB/s eta 0:00:01
   -------------------------------------- - 317.4/332.6 kB 2.2 MB/s eta 0:00:01
   ---------------------------------------- 332.6/332.6 kB 2.1 MB/s eta 0:00:00
Downloading PyArabic-0.6.15-py3-none-any.whl (126 kB)
   ---------------------------------------- 0.0/126.4 kB ? eta -:--:--
   ---------------------------- ----------- 92.2/126.4 kB 2.6 MB/s eta 0:00:01
   ---------------------------------------- 126.4/126.4 kB 2.5 MB/s eta 0:00:00
Downloading Tashaphyne-0.3.6-py3-none-any.whl (251 kB)
   ---------------------------------------- 0.0/251.5 kB ? eta -:--:--
   ----------------- ---------------------- 112.6/251.5 kB 2.2 MB/s eta 0:00:01
   -------------------------------- ------- 204.8/251.5 kB 2.1 MB/s eta 0:00:01
   ---------------------------------------- 251.5/251.5 kB 2.2 MB/s eta 0:00:00
Downloading aiofiles-24.1.0-py3-none-any.whl (15 kB)
Building wheels for collected packages: pickledb, codernitydb3
  Building wheel for pickledb (setup.py): started
  Building wheel for pickledb (setup.py): finished with status 'done'
  Created wheel for pickledb: filename=pickleDB-1.3.2-py3-none-any.whl size=5307 sha256=104132bbb93fc846574c9be4c58e42d679a1fbbf1a9850ea2a9bd3a3f1bff7e3
  Stored in directory: c:\users\fady\appdata\local\pip\cache\wheels\ad\e2\59\dbb52181bcb1551e623e55c2a715add67a2fdbca7d62632da8
  Building wheel for codernitydb3 (setup.py): started
  Building wheel for codernitydb3 (setup.py): finished with status 'done'
  Created wheel for codernitydb3: filename=codernitydb3-0.6.0-py3-none-any.whl size=59877 sha256=4ce7e2e44fbe5f3a8d65b74a8392b4ffc456975e4be3e6a6cc7e4bbeaacdae8e
  Stored in directory: c:\users\fady\appdata\local\pip\cache\wheels\10\46\58\dbd81132e05ee2236e2a7ff9d8a7ec9d00dcfc92dd1e9e6998
Successfully built pickledb codernitydb3
Installing collected packages: mysam-tagmanager, pyarabic, codernitydb3, aiofiles, tashaphyne, pickledb, libqutrub, arramooz-pysqlite, Arabic-Stopwords, naftawayh, alyahmor, qalsadi
Successfully installed Arabic-Stopwords-0.4.3 aiofiles-24.1.0 alyahmor-0.2 arramooz-pysqlite-0.4.2 codernitydb3-0.6.0 libqutrub-1.2.4.1 mysam-tagmanager-0.4 naftawayh-0.4 pickledb-1.3.2 pyarabic-0.6.15 qalsadi-0.5 tashaphyne-0.3.6
Note: you may need to restart the kernel to use updated packages.
[notice] A new release of pip is available: 24.1 -> 25.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [34]:
```
import qalsadi.lemmatizer
lemmer = qalsadi.lemmatizer.Lemmatizer()
words = ['الجري','تجري','يجرون','جري','يجري']

for word in words :
    print(lemmer.lemmatize(word),end=' ')
print()
print (st.stem('يقول'),end=' ')
print (st.stem('يقولون'),end=' ')
print (st.stem('تقول'),end=' ')
print (st.stem('مقوله'))
```
```
جرة تجرة جرى جرة جرى
يقل يقل تقل قول
```

In [35]:
```
text = text = """هل احتاج إلى ترجمة كي تفهم خطاب الملك؟ اللغة "الكلاسيكية" (الفصحى) موجودة في كل اللغات وكذلك اللغة "الدارجة" .. الفرنسية التي ندرس في المدرسة ليست الفرنسية التي يستخدمها الناس في شوارع باريس .. وملكة بريطانيا لا تخطب بلغة شوارع لندن .. لكل مقام مقال"""
lemmas = lemmer.lemmatize_text(text)
print(lemmas)
```
```
['هل', 'احتاج', 'إلى', 'ترجمة', 'كي', 'تف', 'خطاب', 'ملك', '؟', 'لغة', '"', 'كلاسيكي', '"(', 'فصحى', ')', 'موجود', 'في', 'كل', 'لغة', 'كذلك', 'لغة', '"', 'دارج', '"..', 'فرنسة', 'التي', 'درس', 'في', 'مدرس', 'ليست', 'فرنسة', 'التي', 'استخدم', 'ناس', 'في', 'شوارع', 'باريس', '..', 'ملك', 'بريطاني', 'لا', 'خطب', 'بلغة', 'شوارع', 'أذان', '..', 'كل', 'مقام', 'مقال']
```

In [36]:
```
lemmas = lemmer.lemmatize_text(text, return_pos=True)
print(lemmas)
```
```
[('هل', 'stopword'), ('احتاج', 'verb'), ('إلى', 'stopword'), ('ترجمة', 'noun'), ('كي', 'stopword'), ('تف', 'noun'), ('خطاب', 'noun'), ('ملك', 'noun'), ('؟', 'pounct'), ('لغة', 'noun'), ('"', 'pounct'), ('كلاسيكي', 'noun'), ('"(', 'pounct'), ('فصحى', 'noun'), (')', 'pounct'), ('موجود', 'noun'), ('في', 'stopword'), ('كل', 'stopword'), ('لغة', 'noun'), ('كذلك', 'stopword'), ('لغة', 'noun'), ('"', 'pounct'), ('دارج', 'noun'), ('"..', 'pounct'), ('فرنسة', 'noun'), ('التي', 'stopword'), ('درس', 'verb'), ('في', 'stopword'), ('مدرس', 'noun'), ('ليست', 'stopword'), ('فرنسة', 'noun'), ('التي', 'stopword'), ('استخدم', 'verb'), ('ناس', 'noun'), ('في', 'stopword'), ('شوارع', 'noun'), ('باريس', 'all'), ('..', 'pounct'), ('ملك', 'noun'), ('بريطاني', 'noun'), ('لا', 'stopword'), ('خطب', 'verb'), ('بلغة', 'noun'), ('شوارع', 'noun'), ('أذان', 'verb'), ('..', 'pounct'), ('كل', 'stopword'), ('مقام', 'noun'), ('مقال', 'noun')]
```

In [37]:
```
# put diacritics on text
lemmer.set_vocalized_lemma()
lemmas = lemmer.lemmatize_text(text)
print(lemmas)
```
```
[illegible]
```

In [ ]:
```
#
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

special arabic cleaning functions

In [38]:
```
# special cleaning to arabic text
import re
arabic_diacritics = re.compile("""
                             ّ    | # Tashdid
                             َ    | # Fatha
                             ً    | # Tanwin Fath
                             ُ    | # Damma
                             ٌ    | # Tanwin Damm
                             ِ    | # Kasra
                             ٍ    | # Tanwin Kasr
                             ْ    | # Sukun
                             ـ     # Tatwil/Kashida
                         """, re.VERBOSE)


def normalize_arabic(text):
    text = re.sub("[إأآا]", "ا", text)
    text = re.sub("ى", "ي", text)
    text = re.sub("ؤ", "ء", text)
    text = re.sub("ئ", "ء", text)
    text = re.sub("ة", "ه", text)
    text = re.sub("گ", "ك", text)
    return text


def remove_diacritics(text):
    text = re.sub(arabic_diacritics, '', text)
    return text


def remove_repeating_char(text):
    return re.sub(r'(.)\1+', r'\1', text)

print(' '.join(lemmas))
print('----------------------------------------------------')
print(remove_diacritics(' '.join(lemmas)))
print('----------------------------------------------------')
print(normalize_arabic(remove_diacritics(' '.join(lemmas))))
```
```
[illegible]
----------------------------------------------------
هل احتاج إلى ترجمة كي تفهم خطاب ملك ؟ لغة " كلاسيكي "( فصحى ) موجود في كل لغة كذلك لغة " دارج ".. فرنسي التي درس في مدرسة ليست فرنسي التي استخدم ناس في شوارع باريس .. ملك بريطانيا لا خطب بلغة شوارع أذان .. كل مقام مقال
----------------------------------------------------
هل احتاج الى ترجمه كي تفهم خطاب ملك ؟ لغه " كلاسيكي "( فصحى ) موجود في كل لغه كذلك لغه " دارج ".. فرنسي التي درس في مدرسه ليست فرنسي التي استخدم ناس في شوارع باريس .. ملك بريطانيا لا خطب بلغه شوارع اذان .. كل مقام مقال
```

In [ ]: