Working of PorterStemmer

```
from nltk.stem import PorterStemmer

stemmer = PorterStemmer()

print (stemmer.stem('working'))
print (stemmer.stem('works'))
print (stemmer.stem('worked'))
```

```
work
work
work
```

Different types of Tokenizers available.

```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

```
True
```

```
from nltk.tokenize import (word_tokenize,
                           wordpunct_tokenize,
                           TweetTokenizer,
                           MWETokenizer)

text_sample = "It's true, Ms. Priyanka Jones! #Truth"

print(f'Whitespace tokenization = {text_sample.split()}')
print(f'Punctuation-based tokenization =
{wordpunct_tokenize(text_sample)}')
tokenizer = MWETokenizer()
tokenizer.add_mwe(('Priyanka', 'Jones'))
print(f'Multi-word expression (MWE) tokenization =
{tokenizer.tokenize(word_tokenize(text_sample))}')
tokenizer = TweetTokenizer()
print(f'Tweet-rules based tokenization =
{tokenizer.tokenize(text_sample)}')
```

```
Whitespace tokenization = ["It's", 'true,', 'Ms.', 'Priyanka',
'Jones!', '#Truth']
Punctuation-based tokenization = ['It', "'", 's', 'true', ',', 'Ms',
'.', 'Priyanka', 'Jones', '!', '#', 'Truth']
Multi-word expression (MWE) tokenization = ['It', "'s", 'true', ',',
'Ms.', 'Priyanka_Jones', '!', '#', 'Truth']
Tweet-rules based tokenization = ["It's", 'true', ',', 'Ms', '.',
'Priyanka', 'Jones', '!', '#Truth']
```

PorterStemmer for a Sentence.

```
from nltk.stem import PorterStemmer

# tokenize text
text = "A quick brown fox jumps over the lazy dog."
text = text.lower()
words = word_tokenize(text)

print ("word_tokenize: ",words)

########################################################## Way 1
#########################################################################
#
stemmer = PorterStemmer()

words_stem = [stemmer.stem(word) for word in words]

print (words_stem)
```

```
word_tokenize:  ['a', 'quick', 'brown', 'fox', 'jumps', 'over', 'the',
'lazy', 'dog', '.']
['a', 'quick', 'brown', 'fox', 'jump', 'over', 'the', 'lazi', 'dog',
'.']
```

```
########################################################## Way 2
#########################################################################
#
# The above line of code is a shorter version of the following code:
words_stem = []

for word in words:
     words_stem.append(stemmer.stem(word))


words_stem_2 = [str(item) for item in words_stem]
print (words_stem_2)
```

```
['a', 'quick', 'brown', 'fox', 'jump', 'over', 'the', 'lazi', 'dog',
'.']
```

```
########################################################## Way 3
#########################################################################
#
#Coverting back to a sentence.
stemmer = PorterStemmer()
text = "A quick brown fox jumps over the lazy dog."
text_stem = " ".join([stemmer.stem(word) for word in text.split()])
print (text_stem)
```

```
A quick brown fox jump over the lazi dog.
```

SnowballStemmer is another very useful stemming algorithm.

It supports 15 non-English languages. In order to use this steaming class, we need to create an instance with the name of the language we are using and then call the stem() method.

it is a better version of the Porter Stemmer and is more aggressive than Porter Stemmer.

```
from nltk.stem import SnowballStemmer

# Languages supported by SnowballStemmer
print (SnowballStemmer.languages)
```

```
('arabic', 'danish', 'dutch', 'english', 'finnish', 'french',
'german', 'hungarian', 'italian', 'norwegian', 'porter', 'portuguese',
'romanian', 'russian', 'spanish', 'swedish')
```

```
stemmer_english = SnowballStemmer('english')

print (stemmer_english.stem('working'))
print (stemmer_english.stem('works'))
print (stemmer_english.stem('worked'))
```

```
work
work
work
```

Comparison of PorterStemmer and SnowballStemmer

```
p_stemmer = PorterStemmer()
words = ['run','runner','running','ran','runs','easily','fairly']
for word in words:
    print(word+' --> '+p_stemmer.stem(word))
```

```
run --> run
runner --> runner
running --> run
ran --> ran
runs --> run
easily --> easili
fairly --> fairli
```

Note how the stemmer recognizes “runner” as a noun, not a verb form or participle. Also, the adverbs “easily” and “fairly” are stemmed to the unusual root “easili” and “fairli”

```
s_stemmer = SnowballStemmer('english')
words = ['run','runner','running','ran','runs','easily','fairly']
for word in words:
    print(word+' --> '+s_stemmer.stem(word))
```

```
run --> run
runner --> runner
running --> run
```

```
ran --> ran
runs --> run
easily --> easili
fairly --> fair
```

In this case, the stemmer performed the same as the Porter Stemmer, with the exception that it handled the stem of “fairly” more appropriately with “fair”

Lemmatizer

```
import nltk
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Unzipping corpora/wordnet.zip.
```

```
True
```

```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

# Lemmatisation depends upon the Part of Speech of the word
# lemmatize(word, pos=NOUN)
# the default part of speech (pos) for lemmatize method is "n", i.e.
noun
# we can specify part of speech (pos) value like below:
# noun = n, verb = v, adjective = a, adverb = r

print (lemmatizer.lemmatize('is'))
print (lemmatizer.lemmatize('are'))

print (lemmatizer.lemmatize('is', pos='v'))
print (lemmatizer.lemmatize('are', pos='v'))

print (lemmatizer.lemmatize('working', pos='n'))
print (lemmatizer.lemmatize('working', pos='v'))
```

```
is
are
be
be
working
work
```

Lemmatizer for a sentence.

```
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

text = "A quick brown fox jumps over the lazy dog."
```

```
# Normalize text
# NLTK considers capital letters and small letters differently.
# For example, Fox and fox are considered as two different words.
# Hence, we convert all letters of our text into lowercase.
text = text.lower()

# tokenize text
words = word_tokenize(text)

print (words)

############################################################ Way 1
#########################################################################
#
lemmatizer = WordNetLemmatizer()

words_lemma = [lemmatizer.lemmatize(word) for word in words]


print (words_lemma)
```

```
['a', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog',
'.']
['a', 'quick', 'brown', 'fox', 'jump', 'over', 'the', 'lazy', 'dog',
'.']
```

```
########################################################## Way 2
#########################################################################
#
# The above line of code is a shorter version of the following code:

words_lemma = []

for word in words:
     words_lemma.append(lemmatizer.lemmatize(word))


words_lemma_2 = [str(item) for item in words_lemma]
print (words_lemma_2)
```

```
['a', 'quick', 'brown', 'fox', 'jump', 'over', 'the', 'lazy', 'dog',
'.']
```

```
# import modules
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()

sentence = "There are more planets than stars in our galaxy. The
```

```
current count orbiting our star: eight."
punctuations="?:!.,;"
words = nltk.word_tokenize(sentence)

# remove punctuations
for word in words:
    if word in punctuations:
        words.remove(word)

print("Word","     ","Lemma")
for word in words:
    print (word,"---->",lemmatizer.lemmatize(word))
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
Word       Lemma
There ----> There
are ----> are
more ----> more
planets ----> planet
than ----> than
stars ----> star
in ----> in
our ----> our
galaxy ----> galaxy
The ----> The
current ----> current
count ----> count
orbiting ----> orbiting
our ----> our
star ----> star
eight ----> eight
```

Question: Create a text file in local, read the contents of the file, perform stemming and lemmatization and store the results in another file.