Code:

import re

import nltk

import pke

from nltk.tokenize import sent\_tokenize

import contractions

from nltk.tokenize import word\_tokenize

from bs4 import BeautifulSoup

from word2number import w2n

# silly test html

raw\_html = """<a href='#'>hi</a>            <b>.hello</b>. This is us. You'll. One million"""

cleantext = BeautifulSoup(raw\_html, "lxml").text

def removeWhiteSpaces(sentence):

    sentence = " ".join(re.split("\s+", sentence, flags=re.UNICODE))

    return sentence

def getSentences(text):

    return sent\_tokenize(text)

def removeContractions(text):

    newText = list()

    for sentece in text:

        newText.append(contractions.fix(sentece))

    return newText

def removeSpecialChars(text):

    newText = list()

    for sentence in text:

        newText.append(re.sub('[^A-Za-z0-9]+', ' ', sentence))

    return newText

def toLowerCase(text):

    newText = list()

    for sentence in text:

        newText.append(sentence.lower())

    return newText

def textTokenize(text):

    newList = list()

    for sentence in text:

        newList.append(word\_tokenize(sentence))

    return newList

def keyPhraseExtraction(cleantext):

    # initialize keyphrase extraction model, here TopicRank

    extractor = pke.unsupervised.TopicRank()

    extractor.load\_document(

        cleantext,

        language='en',

        normalization='stemming')

    # select the keyphrase candidates, for TopicRank the longest sequences of

    # nouns and adjectives

    extractor.candidate\_selection(pos={'NOUN', 'PROPN', 'ADJ'})

    # weight the candidates using a random walk. The threshold parameter sets the

    # minimum similarity for clustering, and the method parameter defines the

    # linkage method

    extractor.candidate\_weighting(threshold=0.74,

                                method='average')

    keyPhraseList = list()

    # print the n-highest (10) scored candidates

    for (keyphrase, score) in extractor.get\_n\_best(n=1, stemming=True):

        print(keyphrase, score)

        keyPhraseList.append([keyphrase,score])

    return keyPhraseList

def convertToNumbers(text):

    newText = list()

    for test\_str in text:

        # Convert numeric words to numbers

        # Using word2number

        try:

            res = w2n.word\_to\_num(test\_str)

        except:

            res = test\_str

        newText.append(str(res))

    return newText

def removeNumbers(text):

    newText = list()

    for ini\_string in text:

        # using join and isdigit

        # to remove numeric digits from string

        res = ''.join([i for i in ini\_string if not i.isdigit()])

        newText.append(res)

    return newText

# initializing string

print("after removing HTML tags:"+cleantext)

text = removeWhiteSpaces(cleantext)

print("after removing extra whitespace:"+text)

text = getSentences(text)

print("getting sentences:"+str(text))

text = removeContractions(text)

print("removing contractions:"+str(text))

text = removeSpecialChars(text)

print("removing special chars:"+str(text))

text = toLowerCase(text)

print("lowering case:"+str(text))

text = textTokenize(text)

print("tokenizing text:"+str(text))

text = convertToNumbers(text)

print("converting text to numbers:"+str(text))

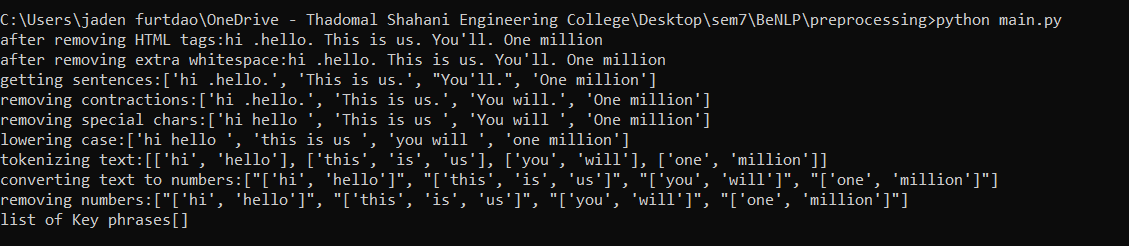
text = removeNumbers(text)

print("removing numbers:"+str(text))

listOfKeyPhrases = keyPhraseExtraction(cleantext)

print("list of Key phrases"+str(listOfKeyPhrases))

Output:



Code:

#3. Apply various other text pre-processing techniques for any given text : Stop Word  Removal, Lemmatization / Stemming

from nltk.tokenize import word\_tokenize

from nltk.corpus import stopwords

import nltk

from nltk.stem import PorterStemmer

from nltk.stem import WordNetLemmatizer

ps = PorterStemmer()

sentence = "Books are on the table"

# tokenization

words = word\_tokenize(sentence)

def removeStopWords(text):

    stop\_words = set(stopwords.words('english'))

    newSentences = list()

    for sentence in text:

        word\_tokens = word\_tokenize(sentence)

        filtered\_sentence = [w for w in word\_tokens if not w in stop\_words]

        newSentences.append(filtered\_sentence)

    return newSentences

# stemming

def stemmingWords(sentence):

    stemmedWords = list()

    for word in sentence.split():

        stemmedWords.append(ps.stem(word))

    return stemmedWords

# Lemmatization:

def lemmatizeWords(words):

    lemmatizer = WordNetLemmatizer()

    newWords = list()

    for word in words:

        wordNoun = lemmatizer.lemmatize("Machine", pos='n')

        # pos: parts of speech tag, verb

        wordVerb = lemmatizer.lemmatize("caring", pos='v')

        newWords.append([wordNoun,wordVerb])

    return newWords

print("Before pre processing:"+sentence)

print("After stemming:"+str(stemmingWords(sentence)))

print("After lemmatization:"+str(lemmatizeWords(words)))

Output:

