3. NLTK

March 29, 2022

[1]:
```
#!pip install nltk
```

[2]:
```
import nltk
#nltk.download()           # only first time after installing nltk; a dialog␣
 ↪box will open in diff window
```

[3]:
```
#. 1. Tokenize text and/or remove nouns, etc. from text by POS tagging
#  2. Convert text to lowercase
#  3. Remove full stops, commas, hyphens, question marks, exclamations, etc.
#  4. Tokenize text
#  5. Remove contractions e.g. isn't = is not
#  6. Remove stop words like the, and, a, is, ...
#  7. Lemmatize i.e. find base words for words in different forms e.g.␣
 ↪lemma(cities) = city
```

[4]:
```
text="""Hello Mr. Smith, how're you doing today? The weather is great, and city␣
 ↪is awesome.
The sky is pinkish-blue. You shouldn't be eating cardboard"""
```

[5]:
```
# Sentence Tokenization
from nltk.tokenize import sent_tokenize
sentences = sent_tokenize(text)
sentences
```

[5]:
```
["Hello Mr. Smith, how're you doing today?",
 'The weather is great, and city is awesome.',
 'The sky is pinkish-blue.',
 "You shouldn't be eating cardboard"]
```

[6]:
```
# Word Tokenization with POS (part-of-speech) tagging
# e.g. to remove proper nouns (NNP) and plural proper nouns (NNPS) from text
# Do this before converting text to lowercase
from nltk.tag import pos_tag
tagged_words = pos_tag(text.split())
non_propernouns = [word for word,pos in tagged_words if pos!='NNP' and pos!
 ↪='NNPS']
non_propernouns
```

[6]:
```
["how're",
 'you',
 'doing',
 'today?',
 'The',
 'weather',
 'is',
 'great,',
 'and',
 'city',
 'is',
 'awesome.',
 'The',
 'sky',
 'is',
 'pinkish-blue.',
 'You',
 "shouldn't",
 'be',
 'eating',
 'cardboard']
```

[7]:
```
# Tokenization without POS tagging

# Removing special characters
import string
text = text.lower()
text = text.replace(".","")
text = text.replace(",","")
text = text.replace("?","")
text = text.replace("-"," ")
text = text.replace("!"," ")
print(text)

# Word Tokenization
words = text.split()
words
```

```
hello mr smith how're you doing today the weather is great and city is awesome
the sky is pinkish blue you shouldn't be eating cardboard
```

[7]:
```
['hello',
 'mr',
 'smith',
 "how're",
 'you',
 'doing',
```

```
 'today',
 'the',
 'weather',
 'is',
 'great',
 'and',
 'city',
 'is',
 'awesome',
 'the',
 'sky',
 'is',
 'pinkish',
 'blue',
 'you',
 "shouldn't",
 'be',
 'eating',
 'cardboard']
```

[8]:
```
contractions = {
"ain't": "is not","aren't": "are not","can't": "cannot","can't've": "cannot␣
↪have","'cause": "because",
"could've": "could have","couldn't": "could not","couldn't've": "could not␣
↪have","didn't": "did not",
"doesn't": "does not","don't": "do not","hadn't": "had not","hadn't've": "had␣
↪not have","hasn't": "has not",
"haven't": "have not","he'd": "he would","he'd've": "he would have","he'll":␣
↪"he will","he'll've": "he he will have",
"he's": "he is","how'd": "how did","how'd'y": "how do you","how'll": "how␣
↪will","how're": "how are","how's": "how is",
"I'd": "I would", "I'd've": "I would have","I'll": "I will","I'll've": "I will␣
↪have","I'm": "I am","I've": "I have",
"i'd": "i would","i'd've": "i would have","i'll": "i will","i'll've": "i will␣
↪have","i'm": "i am","i've": "i have",
"isn't": "is not","it'd": "it would","it'd've": "it would have","it'll": "it␣
↪will","it'll've": "it will have",
"it's": "it is","let's": "let us","ma'am": "madam","mayn't": "may␣
↪not","might've": "might have","mightn't": "might not",
"mightn't've": "might not have","must've": "must have","mustn't": "must␣
↪not","mustn't've": "must not have",
"needn't": "need not","needn't've": "need not have","o'clock": "of the␣
↪clock","oughtn't": "ought not",
"oughtn't've": "ought not have","shan't": "shall not","sha'n't": "shall␣
↪not","shan't've": "shall not have",
```

```
 "she'd": "she would","she'd've": "she would have","she'll": "she␣
 ↪will","she'll've": "she will have",
 "she's": "she is","should've": "should have","shouldn't": "should␣
 ↪not","shouldn't've": "should not have",
 "so've": "so have","so's": "so as","that'd": "that would","that'd've": "that␣
 ↪would have","that's": "that is",
 "there'd": "there would","there'd've": "there would have","there's": "there␣
 ↪is","they'd": "they would",
 "they'd've": "they would have","they'll": "they will","they'll've": "they will␣
 ↪have","they're": "they are",
 "they've": "they have","to've": "to have","wasn't": "was not","we'd": "we␣
 ↪would","we'd've": "we would have",
 "we'll": "we will","we'll've": "we will have","we're": "we are","we've": "we␣
 ↪have","weren't": "were not",
 "what'll": "what will","what'll've": "what will have","what're": "what␣
 ↪are","what's": "what is","what've": "what have",
 "when's": "when is","when've": "when have","where'd": "where did","where's":␣
 ↪"where is","where've": "where have",
 "who'll": "who will","who'll've": "who will have","who's": "who is","who've":␣
 ↪"who have","why's": "why is",
 "why've": "why have","will've": "will have","won't": "will not","won't've":␣
 ↪"will not have","would've": "would have",
 "wouldn't": "would not","wouldn't've": "would not have","y'all": "you␣
 ↪all","y'all'd": "you all would",
 "y'all'd've": "you all would have","y'all're": "you all are","y'all've": "you␣
 ↪all have","you'd": "you would",
 "you'd've": "you would have","you'll": "you will","you'll've": "you will␣
 ↪have","you're": "you are","you've": "you have"
}
```

[9]:
```
words = [contractions.get(n, n) for n in words]
words
```

[9]:
```
['hello',
 'mr',
 'smith',
 'how are',
 'you',
 'doing',
 'today',
 'the',
 'weather',
 'is',
 'great',
 'and',
 'city',
```

```
 'is',
 'awesome',
 'the',
 'sky',
 'is',
 'pinkish',
 'blue',
 'you',
 'should not',
 'be',
 'eating',
 'cardboard']
```

[10]:
```
text = ' '.join([str(word) for word in words])
text
```

[10]:
```
'hello mr smith how are you doing today the weather is great and city is awesome
the sky is pinkish blue you should not be eating cardboard'
```

[11]:
```
words = text.split()
words
```

[11]:
```
['hello',
 'mr',
 'smith',
 'how',
 'are',
 'you',
 'doing',
 'today',
 'the',
 'weather',
 'is',
 'great',
 'and',
 'city',
 'is',
 'awesome',
 'the',
 'sky',
 'is',
 'pinkish',
 'blue',
 'you',
 'should',
 'not',
 'be',
```

```
 'eating',
 'cardboard']
```

[12]:
```
# Removing stopwords
from nltk.corpus import stopwords
# print(stopwords.words('english'))
stop_words = set(stopwords.words('english'))
filtered_words = [w for w in words if not w in stop_words]
filtered_words
```

[12]:
```
['hello',
 'mr',
 'smith',
 'today',
 'weather',
 'great',
 'city',
 'awesome',
 'sky',
 'pinkish',
 'blue',
 'eating',
 'cardboard']
```

[13]:
```
# Stemming: finds stem of given word by chopping of suffix e.g.␣
 ↪stem(cities)=citi
from nltk.stem.porter import PorterStemmer
porter_stemmer = PorterStemmer()
stem_words = [porter_stemmer.stem(w) for w in filtered_words]
stem_words
```

[13]:
```
['hello',
 'mr',
 'smith',
 'today',
 'weather',
 'great',
 'citi',
 'awesom',
 'sky',
 'pinkish',
 'blue',
 'eat',
 'cardboard']
```

[14]:
```
# Lemmatization: better than stemming; finds actual base word e.g.␣
 ↪lemma(cities)=city
```

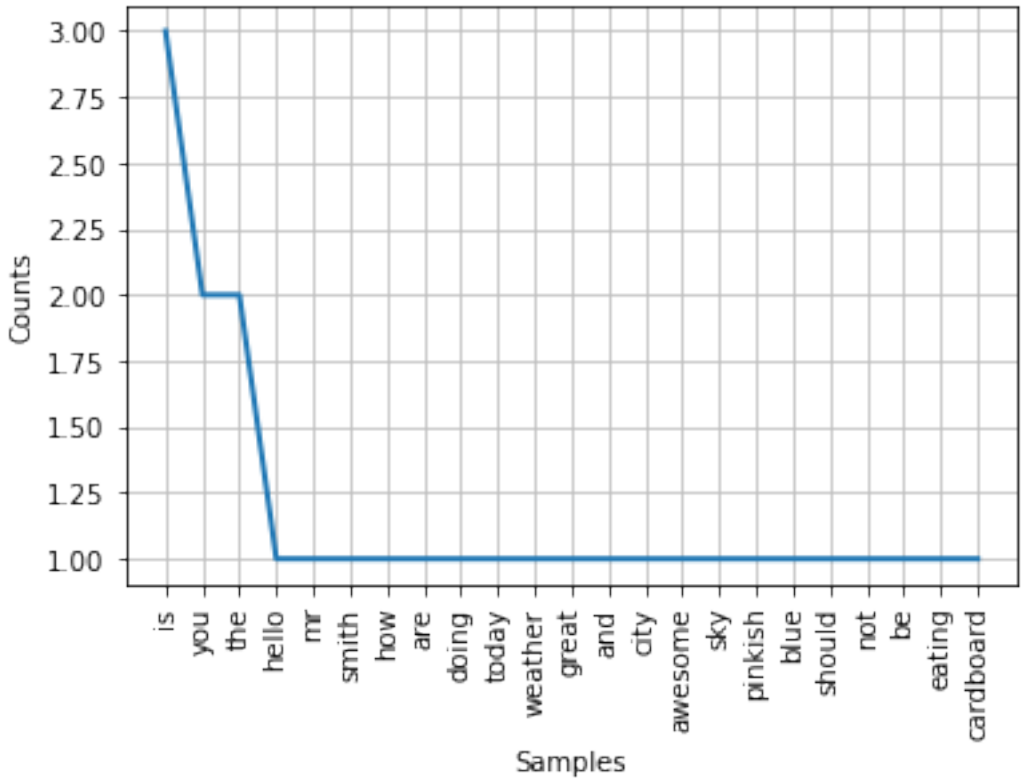
```
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
lem_words = [lemmatizer.lemmatize(w) for w in filtered_words]
lem_words
```

[14]:
```
['hello',
 'mr',
 'smith',
 'today',
 'weather',
 'great',
 'city',
 'awesome',
 'sky',
 'pinkish',
 'blue',
 'eating',
 'cardboard']
```

[15]:
```
# Frequency distribution
from nltk.probability import FreqDist
fdist = FreqDist(words)
fdist.most_common(2)
```

[15]:
```
[('is', 3), ('you', 2)]
```

[16]:
```
# Frequency Distribution Plot
import matplotlib.pyplot as plt
fdist.plot(30,cumulative=False)
plt.show()
```



[17]:
```
# Bigrams and Trigrams

#import nltk
raw = "how are you doing are you doing are"

tokens = nltk.word_tokenize(raw)

#Create your bigrams
bgs = nltk.bigrams(tokens)
#compute frequency distribution for all the bigrams in the text
fdist = nltk.FreqDist(bgs)
fdist
```

[17]:
```
FreqDist({('are', 'you'): 2, ('you', 'doing'): 2, ('doing', 'are'): 2, ('how',
'are'): 1})
```

[18]:
```
#Create your trigrams
bgs = nltk.trigrams(tokens)
#compute frequency distribution for all the bigrams in the text
fdist = nltk.FreqDist(bgs)
fdist
```

[18]:
```
FreqDist({('are', 'you', 'doing'): 2, ('you', 'doing', 'are'): 2, ('how', 'are',
'you'): 1, ('doing', 'are', 'you'): 1})
```

[19]:
```
import collections
from nltk import ngrams

sentence = 'how are you doing how are you doing'

n = 4
sixgrams = ngrams(sentence.split(), n)

sixgramFreq = collections.Counter(sixgrams)
sixgramFreq
```

[19]:
```
Counter({('how', 'are', 'you', 'doing'): 2,
         ('are', 'you', 'doing', 'how'): 1,
         ('you', 'doing', 'how', 'are'): 1,
         ('doing', 'how', 'are', 'you'): 1})
```

[ ]:
```

```