Homework-0

Achyut Ganti

- Link to Google Colab notebook
- Link to Dataset. I removed the Foreword and Index.
- GitHub Repo

Motivation

The program takes a plain text file as an input and outputs a list of each normalized token by the number of times they appear in the file, largest to smallest. To options to normalize/pre-process our data include **casefolding**, **punctuation removal**, **stopwords**, **stemming** and **lemmatization**

The order in which we apply these options are:

1. casefolfing
2. punctuation removal
3. stopwords
4. stemming
5. lemmatization

Code

The first step is to open and read our text file, where the data is stored.

```
newlist = []
# opening our text file
with open('mytext.txt','r') as file:
    for line in file:
        for word in line.split():
            newlist.append(word)
```

The above piece of code opens `mytext.txt` and appends each word in the document to an empty list named `newlist`.

Then begins pre-processing. The first function we build is the casefolding function. We want our document to be lowercased. By applying this function first, we make sure that the entire text is consistent.

The next operation to be applied is punctuation removal. This is helpful for end of the sentence tokens that end with a period. Although it should be noted that it isn't the best way to handle this usecase because the token *Dr.David* will be normalized to *DrDavid* which may not be desirable in some cases.

Then we remove stopwords. These are a set of commonly used words in English and are considered unimportant as they don't add much meaning to the model. We use `nltk's` stopword removal function to handle this preprocessing step.

Next are stemming and lemmatization. They are both used to generate root form of the given word. But lemmatization overcomes a shortcoming faced by stemming which is, it generates actual words which may not happen when we use stemming.

The logic behind using the stopword removal function before stemming/lemmatization is that stopwords are generally short and don't have a separate root word, like "the", "is", "are", "and" etc. Once we casefold our text, we run the stopword function to filter the unwanted words out, and then perform lemmatization on our text..

Function to Casefold

The built-in function used here is `lower()`

```
def loww(newlist):
    lowered = [item.lower() for item in newlist]
    return lowered
```

Function to remove punctuations

```
import string
```

```
def punct(newlist):
    punct = [i.translate(str.maketrans('', '', string.punctuation)) for i in newlist]
    return punct
```

Function to remove stopwords

```
from nltk.corpus import stopwords
nltk.download('stopwords')
stops = set(stopwords.words('english'))

def stop(newlist):
    stopped = [item for item in newlist if item not in stops]
    return stopped
```

Function for stemming the text

```
from nltk.stem import PorterStemmer
ps = PorterStemmer()
```

```
def stem(newlist):
    stemmed = [ps.stem(i) for i in newlist]
    return stemmed
```

Function to lemmatize the text

```
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()

def lemma(newlist):
    lemm = [lemmatizer.lemmatize(i) for i in newlist]
    return lemm
```

Checking for user-defined arguments

```
if "loww" in args:
    newlist = loww(newlist)
  if "punct" in args:
    newlist = punct(newlist)
  if "stop" in args:
    newlist = stop(newlist)
  if "stem" in args:
    newlist = stem(newlist)
  if "lemma" in args:
    newlist = lemma(newlist)
```

Creating a counter to output the tokens and their occurances after the pre-processing stage

```
newlist = Counter(newlist)
  return sorted(newlist.items(), key=lambda x:x[1], reverse=True)
```

Full example

```
def normalized(newlist, *args):
  def loww(newlist):
    lowered = [item.lower() for item in newlist]
    return lowered
  def stop(newlist):
    stopped = [item for item in newlist if item not in stops]
    return stopped
  def stem(newlist):
    stemmed = [ps.stem(i) for i in newlist]
    return stemmed
  def lemma(newlist):
    lemm = [lemmatizer.lemmatize(i) for i in newlist]
    return lemm
  def punct(newlist):
    punct = ss = [i.translate(str.maketrans('', '', string.punctuation)) for i in newlist]
    return punct
  if "loww" in args:
    newlist = loww(newlist)
  if "punct" in args:
    newlist = punct(newlist)
  if "stop" in args:
    newlist = stop(newlist)
  if "stem" in args:
    newlist = stem(newlist)
  if "lemma" in args:
    newlist = lemma(newlist)

  newlist = Counter(newlist)
  return sorted(newlist.items(), key=lambda x:x[1], reverse=True)
```

Testing

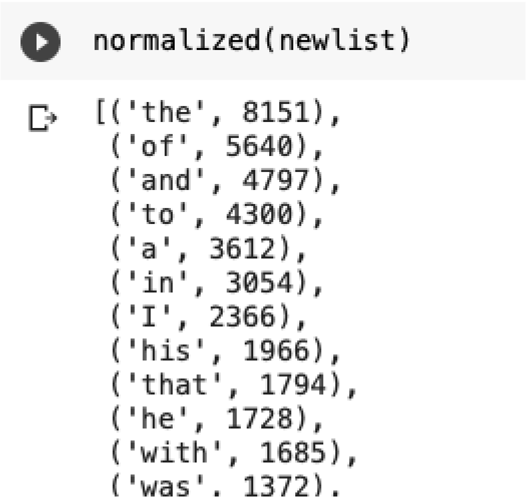
The function `normalized(newlist, *args)` takes a text file and `*args` as inputs. The options for `*args` are as follows:
`loww` for casefolding
`stop` for stopword removal
`stem` for stemming
`lemma` for lemmatization of text
`punct` for puncutation removal

For example `normalized(newlist, "loww", "stem"` will first change the casing to lowercase and then apply the `porter.stemmer` module on the input file.

Similarly `normalized(newlist, "loww","lemma","punct"` will perform casefolding, punctuation removal and lemmatization on the input file and return the final list.

Some examples

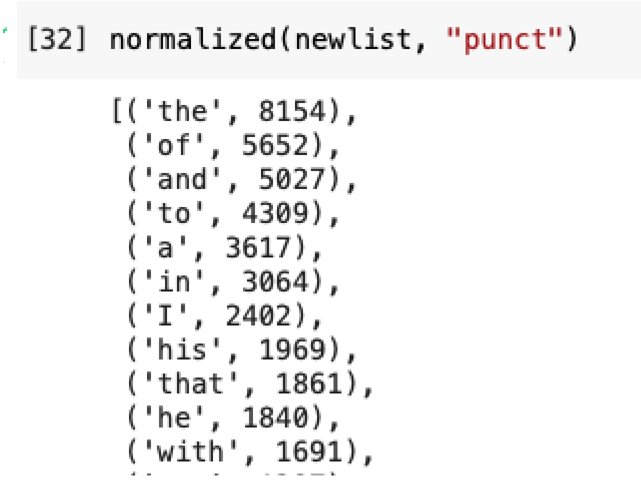
1.Here are tokens in the text without any pre-processing options added



```
normalized(newlist)

[('the', 8151),
 ('of', 5640),
 ('and', 4797),
 ('to', 4300),
 ('a', 3612),
 ('in', 3054),
 ('I', 2366),
 ('his', 1966),
 ('that', 1794),
 ('he', 1728),
 ('with', 1685),
 ('was'. 1372).
```

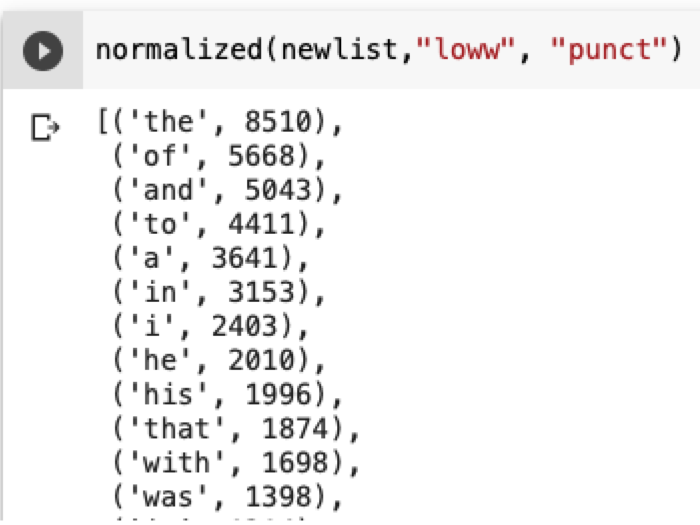
2. Performing only punctuation removal increases the number of tokens in all cases as expected, except for the token "the" which is unusual.



```
[32] normalized(newlist, "punct")

     [('the', 8154),
      ('of', 5652),
      ('and', 5027),
      ('to', 4309),
      ('a', 3617),
      ('in', 3064),
      ('I', 2402),
      ('his', 1969),
      ('that', 1861),
      ('he', 1840),
      ('with', 1691),
```

3. Applying lowercasing and punctuation removal has similar on the text as Case-2

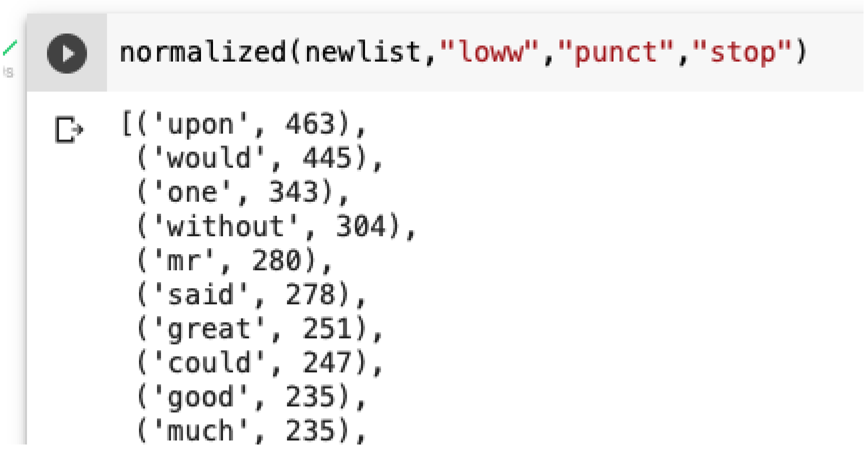


```
normalized(newlist,"loww", "punct")

[('the', 8510),
 ('of', 5668),
 ('and', 5043),
 ('to', 4411),
 ('a', 3641),
 ('in', 3153),
 ('i', 2403),
 ('he', 2010),
 ('his', 1996),
 ('that', 1874),
 ('with', 1698),
 ('was', 1398),
```

4. Fourth stage is applying lowercasing, punctuation and stopword removal. The order of options by the user doesn't matter. They are applied as per the algorithm's order.

We notice that many words in the list that have appeared in previous attempts are missing. This is due to stopword removal technique.



```
normalized(newlist,"loww","punct","stop")
```

```
[('upon', 463),
 ('would', 445),
 ('one', 343),
 ('without', 304),
 ('mr', 280),
 ('said', 278),
 ('great', 251),
 ('could', 247),
 ('good', 235),
 ('much', 235),
```