Double-click (or enter) to edit

Perform the following on the cat text documents:
a. Tokenize
b. Stop words removal
c. stemming/lemmatization
d. Print words with max 5 TFs
e. Print words with min 5 IDFs
f. Find TF-IDF
g. Make histogram of TF-IDF

Double-click (or enter) to edit

```
pip install nltk
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: nltk in /usr/local/lib/python3.7/dist-packages (3.7)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from nltk) (4.64.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from nltk) (1.2.0)
Requirement already satisfied: click in /usr/local/lib/python3.7/dist-packages (from nltk) (7.1.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.7/dist-packages (from nltk) (2022.6.2)
```

```
import nltk
nltk.download('punkt')
nltk.download('stopwords')
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
```

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

```
text = "The cat is a domestic animal. Its scientific name is Felis catus. It is a small animal t
```

a. Tokenize

```
tokenize=word_tokenize(text)
tokenize
```

```
['The',
 'cat',
 'is',
 'a',
 'domestic',
 'animal',
 '.',
 'Its',
 'scientific',
 'name',
 'is',
 'Felis',
 'catus',
 '.',
 'It',
 'is',
 'a',
 'small',
 'animal',
 'that',
 'belongs',
 'to',
 'the',
 '“',
 'Felidae',
 '”',
 'family',
 '.',
 'The',
 'cat',
 'is',
 'the',
 'only',
```

```
 'domesticated',
 'species',
 'of',
 'the',
 'family',
 '.',
 'Other',
 'members',
 'include',
 'tigers',
 ',',
 'panthers',
 ',',
 'etc',
 '.',
 'Cats',
 'are',
 'adorable',
 'animals',
 'and',
 'are',
 'petted',
 'by',
 'lots',
 'of',
```

b. stop words removal

```
#stopwords are removed.
nostop_words = [word for word in tokenize if word not in stopwords.words('english')]
nostop_words
```

```
['The',
 'cat',
 'domestic',
 'animal',
 '.',
 'Its',
 'scientific',
 'name',
 'Felis',
 'catus',
 '.',
 'It',
 'small',
 'animal',
 'belongs',
 '“',
 'Felidae',
 '”',
 'family',
 '.',
 'The',
 'cat',
 'domesticated',
 'species',
 'family',
 '.',
 'Other',
 'members',
 'include',
 'tigers',
 ',',
 'panthers',
 ',',
 'etc',
 '.',
 'Cats',
 'adorable',
 'animals',
 'petted',
 'lots',
 'people',
 'world',
 '.',
 'They',
 'playful',
 'spending',
 'time',
 'reduces',
 'stress',
 'anxiety',
 '.']
```

c. stemming/lemmatization

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

```
nltk.download('wordnet')
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
True
```

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

```
for word in nostop_words:
  r=le.lemmatize(word)
  print(r)
```

```
The
cat
domestic
animal
.
Its
scientific
name
Felis
catus
.
It
small
animal
belongs
“
Felidae
”
family
.
The
cat
domesticated
specie
family
.
Other
member
include
tiger
,
panther
,
etc
.
Cats
adorable
animal
petted
lot
people
world
.
They
playful
spending
time
reduces
stress
anxiety
.
```

```
for word in nostop_words:
  r=ps.stem(word)
  print(r)
```

```
the
cat
domest
anim
.
```

```
it
scientif
name
feli
catu
.
it
small
anim
belong
“
felida
”
famili
.
the
cat
domest
speci
famili
.
other
member
includ
tiger
,
panther
,
etc
.
cat
ador
anim
pet
lot
peopl
world
.
they
play
spend
time
reduc
stress
anxieti
.
```

d.Print words with TFs

```
sent1="""Cats are of three types- house cats, farm cats and feral cats. House cats are the cats
Cats become good friends of humans. Unlike dogs, cats are not very active around their owners.
However, they are good emotional companions to their owners.
An essay on cats must emphasize the fact that cat-sitting has been proven to be therapeutic by m
Cats are omnivores.They eat vegetative items such as rice, milk, pulses, etc. as well as fish, m
Therefore, cats can feed on both types of food."""

sent2 = "Cats have two eyes, a tiny nose, two perky ears, four legs and a tail. Their bodies are

tokenize1 = word_tokenize(sent1)
list_of_words1 = [word for word in tokenize1 if word not in stopwords.words('english')]
list_of_words1
```

```
['Cats',
 'three',
 'types-',
 'house',
 'cats',
 ',',
 'farm',
 'cats',
 'feral',
 'cats',
 '.',
 'House',
 'cats',
 'cats',
 'pet',
 'houses',
 '.',
 'Cats',
 'become',
```

```
'good',
'friends',
'humans',
'.',
'Unlike',
'dogs',
',',
'cats',
'active',
'around',
'owners',
'.',
'However',
',',
'good',
'emotional',
'companions',
'owners',
'.',
'An',
'essay',
'cats',
'must',
'emphasize',
'fact',
'cat-sitting',
'proven',
'therapeutic',
'many',
'researchers',
'.',
'Cats',
'omnivores.They',
'eat',
'vegetative',
'items',
'rice',
',',
'milk'.
```

```
tokenize2 = word_tokenize(sent2)
list_of_words2 = [word for word in tokenize2 if word not in stopwords.words('english')]
list_of_words2
```

```
'animals',
'.',
'They',
'sleep',
'lot',
'day',
'.',
'Cats',
'good',
'friends',
'humans',
```

```
'.',
'Cats',
'witty',
'animals',
'.',
'They',
'skilful',
'hunters',
'rats',
',',
'snakes',
',',
'etc',
' ']
```

```
#Finding unique words
unique_words=set(list_of_words1).union(set(list_of_words2))
unique_words
```

```
'farm',
'feed',
'feral',
'fish',
'food',
'four',
'friends',
'fur',
'globe',
'good',
'house',
'houses',
'humans',
'hunters',
'items',
'kinds',
'lazy',
'legs',
'like',
'lot',
'many',
'meat',
'mice',
'milk',
'must',
'namely-',
'nose',
'omnivores.They',
'owners',
'paws',
'people',
'perky',
'pet',
'petted',
'proven',
'pulses',
'rats',
'researchers',
'rice',
'sacred',
'sharp',
'skilful',
'sleep',
'smooth',
'snakes',
'tail',
'therapeutic',
'three',
'tiny',
'traditions',
'two',
'types',
'types-',
'vegetables.Cats',
'vegetative',
'well',
'whiskers',
'witty'}
```

```
words1 = dict.fromkeys(unique_words,0)
for word in list_of_words1:
  words1[word] += 1
words2 = dict.fromkeys(unique_words,0)
for word in list_of_words1:
  words1[word] += 1
```

```
#Function for computing TF scores
def computeTF(word_dict, list_of_words):
  tf_dict = {}
  words_count = len(list_of_words)
  for word, count in word_dict.items():
    tf_dict[word] = count / float(words_count)
  return tf_dict

tf1 = computeTF(words1, list_of_words1)
tf1
```

```
    'feed': 0.024691358024691357,
    'two': 0.0,
    'cats': 0.19753086419753085,
    'Their': 0.0,
    'day': 0.0,
    'items': 0.024691358024691357,
    'fish': 0.024691358024691357,
    'animals': 0.0,
    'types': 0.024691358024691357,
    'like': 0.0,
    'meat': 0.024691358024691357,
    ',': 0.2716049382716049,
    'Unlike': 0.024691358024691357,
    'culture': 0.0,
    'tiny': 0.0,
    'hunters': 0.0,
    'owners': 0.04938271604938271,
    'therapeutic': 0.024691358024691357,
    'across': 0.0,
    'lot': 0.0,
    'They': 0.0,
    'traditions': 0.0,
    'four': 0.0,
    'essay': 0.024691358024691357,
    'namely-': 0.0,
    'witty': 0.0,
    '.': 0.2222222222222222,
    'researchers': 0.024691358024691357,
    'companions': 0.024691358024691357,
    'fur': 0.0,
    'sacred': 0.0,
    'rice': 0.024691358024691357,
    'Therefore': 0.024691358024691357,
    'nose': 0.0,
    'sharp': 0.0,
    'fact': 0.024691358024691357,
    'dogs': 0.024691358024691357,
    'globe': 0.0,
    'omnivores.They': 0.024691358024691357,
    'milk': 0.024691358024691357,
    'whiskers': 0.0,
    'perky': 0.0,
    'proven': 0.024691358024691357,
    'food': 0.024691358024691357,
    'petted': 0.0,
    'skilful': 0.0,
    'eat': 0.024691358024691357,
    'birds': 0.024691358024691357,
    'bodies': 0.0,
    'However': 0.024691358024691357,
    'pulses': 0.024691358024691357,
    'Japanese': 0.0,
    'covered': 0.0,
    'smooth': 0.0,
    'mice': 0.024691358024691357,
    'An': 0.024691358024691357,
    'around': 0.024691358024691357,
    'feral': 0.024691358024691357}
```

```
tf2 = computeTF(words2, list_of_words2)
tf2
```

```
',': 0.0,
'Unlike': 0.0,
'culture': 0.0,
'tiny': 0.0,
'hunters': 0.0,
'owners': 0.0,
'therapeutic': 0.0,
'across': 0.0,
'lot': 0.0,
'They': 0.0,
'traditions': 0.0,
'four': 0.0,
'essay': 0.0,
'namely-': 0.0,
'witty': 0.0,
'.': 0.0,
'researchers': 0.0,
'companions': 0.0,
'fur': 0.0,
'sacred': 0.0,
'rice': 0.0,
'Therefore': 0.0,
'nose': 0.0,
'sharp': 0.0,
'fact': 0.0,
'dogs': 0.0,
'globe': 0.0,
'omnivores.They': 0.0,
'milk': 0.0,
'whiskers': 0.0,
'perky': 0.0,
'proven': 0.0,
'food': 0.0,
'petted': 0.0,
'skilful': 0.0,
'eat': 0.0,
'birds': 0.0,
'bodies': 0.0,
'However': 0.0,
'pulses': 0.0,
'Japanese': 0.0,
'covered': 0.0,
'smooth': 0.0,
'mice': 0.0,
'An': 0.0,
'around': 0.0,
'feral': 0.0}
```

e. Print IDFs

```
#Function for computing IDF
import math
def computeIDF(documents):
  n = len(documents)
  idf_dict=dict.fromkeys(documents[0].keys(),0)
  for document in documents:
    for word, val in document.items():
      if val>=0:
        idf_dict[word] +=1
  for word, val in idf_dict.items():
    idf_dict[word] = math.log(n / float(val))
  return idf_dict

idfs = computeIDF([words1,words2])
idfs
```

```
 'hunters': 0.0,
 'owners': 0.0,
 'therapeutic': 0.0,
 'across': 0.0,
 'lot': 0.0,
 'They': 0.0,
 'traditions': 0.0,
 'four': 0.0,
 'essay': 0.0,
 'namely-': 0.0,
 'witty': 0.0,
 '.': 0.0,
 'researchers': 0.0,
 'companions': 0.0,
 'fur': 0.0,
 'sacred': 0.0,
 'rice': 0.0,
 'Therefore': 0.0,
 'nose': 0.0,
 'sharp': 0.0,
 'fact': 0.0,
 'dogs': 0.0,
 'globe': 0.0,
 'omnivores.They': 0.0,
 'milk': 0.0,
 'whiskers': 0.0,
 'perky': 0.0,
 'proven': 0.0,
 'food': 0.0,
 'petted': 0.0,
 'skilful': 0.0,
 'eat': 0.0,
 'birds': 0.0,
 'bodies': 0.0,
 'However': 0.0,
 'pulses': 0.0,
 'Japanese': 0.0,
 'covered': 0.0,
 'smooth': 0.0,
 'mice': 0.0,
 'An': 0.0,
 'around': 0.0,
 'feral': 0.0}
```

f. find TF-IDF

```
#Function for computing TF-IDF
def computeTFIDF(tf, idfs):
  tfidf = {}
  for word, val in tf.items():
    tfidf[word] = val * idfs[word]
  return tfidf
```

```
import pandas as pd

tfidf1 = computeTFIDF(tf1, idfs)
tfidf2 = computeTFIDF(tf2, idfs)
df = pd.DataFrame([tfidf1, tfidf2])
df
```

	house	must	considered	become	emotional	etc	friends	many	paws	lazy	...
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...

2 rows × 94 columns

g. Histogram

```
import matplotlib.pyplot as plt

#Histogram for TF-IDF
plt.hist(df)
plt.show
```

```
<function matplotlib.pyplot.show(*args, **kw)>
```