Group A
Assignment No. 7

Aim: Test Analytics:

1) Extract Sample document and apply following document preprocessing methods: Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization.
2) Create representation of document by calculating Term Frequency and Inverse Document Frequency.

Soln :

Step 1: Analysing Dataset

The first step in any of the Machine Learning tasks is to analyse the data. So if we look at the dataset, at first glance, we see all the documents with words in English.

Step 2: Extracting Title & Body:

There is no specific way to do this, this totally depends on the problem statement at hand and on the analysis, we do on the dataset.

Filename	Size	Description of the Textfile
sre01.txt	11278	SRE: The Saga Of The Best SRE Game Ever Played! By Josh Renaud
sre02.txt	5862	Solar Realms Elite: The True Story of the Unsung Heroes, by Josh Renaud
sre03.txt	8555	Solar Realms Elite: Ultra's Untold Story by Josh Renaud
sre04.txt	44198	Solar Realms Elite IV: The Confrontation, by Josh Renaud
sre05.txt	20787	Solar Realms Elite V: The Underground, by Josh Renaud
sre06.txt	26731	Solar Realms Elite VI: The Alliance Restored, by Josh Renaud
sre07.txt	23597	Solar Realms Elite 7: Petros, by Josh Renaud
sre08.txt	33170	Solar Realms Elite VIII: Kazik, by Josh Renaud
sre09.txt	26073	Solar Realms Elite IX: Survival of the Fittest, by Josh Renaud
sre10.txt	25725	Solar Realms Elite X: Legacies, by Josh Renaud
sre_feqh.txt	20054	Solar Realms Elite: The Feqh Galaxy, by Josh Renaud
sre_finl.txt	33158	Solar Realms Elite: The Finale by Josh Renaud
sre_sei.txt	20753	Solar Realms Elite: Galaxy Sei, by Josh Renaud
sretrade.txt	9008	The SRE Commerce and Trade Theories, by Josh Renaud
srex.txt	37128	Solar Realms Elite: X1 and X2, by Josh Renaud

There are 15 files for

Now we can find that **folders** give extra / for the root folder, so we are going to remove it.

```
folders[0] = folders[0][:len(folders[0])-1]
```

```
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="13chil.txt">13chil.txt</A>  <tab t
o=T><TD> 8457<BR><TD> The Story of the Sly Fox
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="14.lws">14.lws</A>  <tab to=T><TD>
5261<BR><TD> A Smart Bomb with a Language Parser
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="16.lws">16.lws</A>  <tab to=T><TD>
15294<BR><TD> Two Guys in a Garage, by M. Pshota
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="17.lws">17.lws</A>  <tab to=T><TD>
10853<BR><TD> The Early Days of a High-Tech Start-up are Magic (November
18, 1991) by M. Peshota
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="18.lws">18.lws</A>  <tab to=T><TD>
26624<BR><TD> The Couch, the File Cabinet, and the Calendar, by M. Pesho
ta (December 9, 1991)
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="19.lws">19.lws</A>  <tab to=T><TD>
17902<BR><TD> Engineering the Future of American Technology by M. Peshot
a (January 5, 1992)
<TR VALIGN=TOP><TD ALIGN=TOP><A HREF="20.lws">20.lws</A>  <tab to=T><TD>
13588<BR><TD> What Research and Development Was Always Meant to Be, by
M. Peshota
```

Names and titles variables have the list of all names and titles.

```
names = re.findall('><A HREF="(.*)">', text)
titles = re.findall('<BR><TD> (.*)\n', text)
dataset = []for i in folders:
    file = open(i+"/index.html", 'r')
    text = file.read().strip()
    file.close()    file_name = re.findall('><A HREF="(.*)">', text)
    file_title = re.findall('<BR><TD> (.*)\n', text)

    for j in range(len(file_name)):
        dataset.append((str(i) + str(file_name[j]), file_title[j]))
```

```
1 dataset
```

```
[('/Users/williamscott/Desktop/IR/Assignments/Assignment 2/stories/100west.txt',
  'Going 100 West by 53 North by Jim Prentice (1990)'),
 ('/Users/williamscott/Desktop/IR/Assignments/Assignment 2/stories/13chil.txt',
  'The Story of the Sly Fox'),
 ('/Users/williamscott/Desktop/IR/Assignments/Assignment 2/stories/14.lws',
  'A Smart Bomb with a Language Parser'),
 ('/Users/williamscott/Desktop/IR/Assignments/Assignment 2/stories/16.lws',
  'Two Guys in a Garage, by M. Pshota'),
 ('/Users/williamscott/Desktop/IR/Assignments/Assignment 2/stories/17.lws',
  'The Early Days of a High-Tech Start-up are Magic (November 18, 1991) by M. Peshota'),
 ('/Users/williamscott/Desktop/IR/Assignments/Assignment 2/stories/18.lws',
  'The Couch, the File Cabinet, and the Calendar, by M. Peshota (December 9, 1991)'),
 ('/Users/williamscott/Desktop/IR/Assignments/Assignment 2/stories/19.lws',
  'Engineering the Future of American Technology by M. Peshota (January 5, 1992)'),
 ('/Users/williamscott/Desktop/IR/Assignments/Assignment 2/stories/20.lws',
  'What Research and Development Was Always Meant to Be, by M. Peshota'),
 ('/Users/williamscott/Desktop/IR/Assignments/Assignment 2/stories/3gables.txt',
  'The Adventure of the Three Gables'),
```

simply use a conditional checker to remove it.

```
if c == False:
    file_name = file_name[2:]
    c = True
```

Step 3: Preprocessing

Preprocessing is one of the major steps when we are dealing with any kind of text model.Few mandatory preprocessing are: converting to lowercase, removing punctuation, removing stop words and lemmatization/stemming. In our problem statement, it seems like the basic preprocessing steps will be sufficient.

Lowercase: Numpy has a method that can convert the list of lists to lowercase at once.

```
np.char.lower(data)
```

Stop words

Stop words are the most commonly occurring words that don't give any additional value to the document vector.

```
print(stopwords.words('english'))
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you',
'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'sh
t's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves
t', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'b
ng', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'i
f', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into',
e', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'o
e', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'bo
me', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than',
'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 'r
n', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't
"isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "ne
n't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn',
```

Iterate over all the stop words and not append them to the list if it's a stop word.

```
new_text = ""
for word in words:
    if word not in stop_words:
        new_text = new_text + " " + word
```

Punctuation

Punctuation is the set of unnecessary symbols that are in our corpus documents.

```
symbols = "!\"#$%&()*+-./:;<=>?@[\]^_`{|}~\n"
for i in symbols:
    data = np.char.replace(data, i, ' ')
```

Apostrophe

Note that there is no ' apostrophe in the punctuation symbols

```
return np.char.replace(data, "'", "")
```

Single Characters

Single characters are not much useful in knowing the importance of the document and few final single characters might be irrelevant symbols, so it is always good to remove the single characters.

```
new_text = ""
for w in words:
    if len(w) > 1:
        new_text = new_text + " " + w
```

Stemming

This is the final and most important part of the preprocessing. stemming converts words to their stem.

```
stemmer = PorterStemmer()
stemmer.stem("swimming")
```

```
'swim'
```

Lemmatisation

Lemmatisation is a way to reduce the word to the root synonym of a word.

Stemming vs Lemmatization

stemming — need not be a dictionary word, removes prefix and affix based on few rules

lemmatization — will be a dictionary word. reduces to a root synonym.

Word	***Lemmatization***	***Stemming***
was	be	wa
studies	study	studi
studying	study	study

Converting Numbers

achieve this we are going to use a library called **num2word**.

```
num2words(100500)
```

```
'one hundred thousand, five hundred'
```

Preprocessing

Finally, we are going to put in all those preprocessing methods above in another method and we will call that preprocess method.

```
def preprocess(data):
    data = convert_lower_case(data)
    data = remove_punctuation(data)
    data = remove_apostrophe(data)
    data = remove_single_characters(data)
    data = convert_numbers(data)
    data = remove_stop_words(data)
    data = stemming(data)
    data = remove_punctuation(data)
    data = convert_numbers(data)
```

Step 4: Calculating TF-IDF

Calculating DF

```
DF = {}
for i in range(len(processed_text)):
    tokens = processed_text[i]
    for w in tokens:
        try:
            DF[w].add(i)
        except:
            DF[w] = {i}
```

```
for i in DF:
    DF[i] = len(DF[i])
DF
```

```
{'sharewar': 1,
 'trial': 1,
 'project': 4,
 'freewar': 1,
 'need': 6,
 'support': 2,
 'continu': 4,
 'one': 10,
 'hundr': 8,
```

To find the total unique words in our vocabulary, we need to take all the keys of DF.

```
total_vocab = [x for x in DF]
print(total_vocab)
```

```
['sharewar', 'trial', 'project', 'freewar', 'ne
'north', 'jim', 'prentic', 'copyright', 'thousa
c', 'phrase', 'spoken', 'mumbl', 'thought', 'ir
'map', 'label', 'degr', 'presenc', 'indic', 'hi
ct', 'mind', 'intern', 'border', 'writer', 'poe
le', 'man', 'eat', 'mosquito', 'murder', 'hord
'dog', 'stori', 'record', 'break', 'trout', 'wa
'forest', 'crash', 'moo', 'tear', 'brush', 'tre
uck', 'feed', 'quiet', 'pond', 'placid', 'bay',
r', 'authent', 'northern', 'scene', 'wildlif',
d', 'person', 'live', 'farther', 'becom', 'appa
```

Calculating TF-IDF

```
tf_idf = {}
for i in range(N):
    tokens = processed_text[i]
    counter = Counter(tokens + processed_title[i])
    for token in np.unique(tokens):
        tf = counter[token]/words_count
        df = doc_freq(token)
        idf = np.log(N/(df+1))
        tf_idf[doc, token] = tf*idf
```

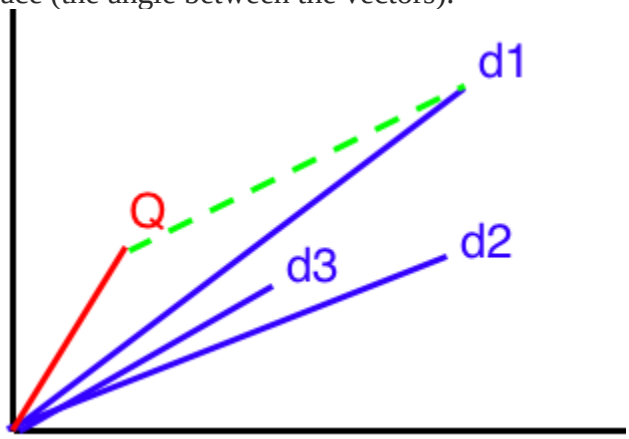
Step 4: Ranking using Matching Score

Matching score is the simplest way to calculate the similarity, in this method, we **add tf_idf values of the tokens that are in query for every document.**

```
def matching_score(query):
    query_weights = {}
    for key in tf_idf:
        if key[1] in tokens:
            query_weights[key[0]] += tf_idf[key]
```

Step 5: Ranking using Cosine Similarity

It will mark all the documents as vectors of tf-idf tokens and measures the similarity in cosine space (the angle between the vectors).



Vectorization

To compute any of the above, the simplest way is to convert everything to a vector and then compute the cosine similarity.

```
# Document Vectorization
D = np.zeros((N, total_vocab_size))
for i in tf_idf:
```

```
    ind = total_vocab.index(i[1])
    D[i[0]][ind] = tf_idf[i]
Q = np.zeros((len(total_vocab)))
counter = Counter(tokens)
words_count = len(tokens)
query_weights = {}
for token in np.unique(tokens):
    tf = counter[token]/words_count
    df = doc_freq(token)
    idf = math.log((N+1)/(df+1))
```

Analysis

Short Query

```
Matching Score

Query: Without the drive of Rebeccah's insistence, Kate lost her momentum. She stood next a slatted oak bench, canist
ers still clutched, surveying

['without', 'drive', 'rebeccah', 'insist', 'kate', 'lost', 'momentum', 'stood', 'next', 'slat', 'oak', 'bench', 'cani
st', 'still', 'clutch', 'survey']

[166, 200, 352, 433, 211, 350, 175, 187, 188, 294]
Cosine Similarity

Query: Without the drive of Rebeccah's insistence, Kate lost her momentum. She stood next a slatted oak bench, canist
ers still clutched, surveying

['without', 'drive', 'rebeccah', 'insist', 'kate', 'lost', 'momentum', 'stood', 'next', 'slat', 'oak', 'bench', 'cani
st', 'still', 'clutch', 'survey']

[200 166 433 175 169 402 211  87 151 369]
```

Long Query

```
Matching Score

Query: And then she recalled his stiff body stretched out in the little bed over the garages. Another pearl had come
loose from the strand, seeming to want to search out its old home in a far away oyster bed. She would have those pear
ls laid out n

['recal', 'stiff', 'bodi', 'stretch', 'littl', 'bed', 'garag', 'anoth', 'pearl', 'come', 'loo', 'strand', 'seem', 'wa
nt', 'search', 'old', 'home', 'far', 'away', 'oyster', 'bed', 'would', 'pearl', 'laid']

[352, 351, 43, 269, 17, 272, 9, 67, 416, 208]
Cosine Similarity

Query: And then she recalled his stiff body stretched out in the little bed over the garages. Another pearl had come
loose from the strand, seeming to want to search out its old home in a far away oyster bed. She would have those pear
ls laid out n

['recal', 'stiff', 'bodi', 'stretch', 'littl', 'bed', 'garag', 'anoth', 'pearl', 'come', 'loo', 'strand', 'seem', 'wa
nt', 'search', 'old', 'home', 'far', 'away', 'oyster', 'bed', 'would', 'pearl', 'laid']

[ 16   3 200 151  65 364 169  87  27  82]
```