clustering-Text-K-MEANS-NLP

March 23, 2024

[1]:
```
 from sklearn.datasets import fetch_20newsgroups
from nltk.tokenize import word_tokenize #Used to extract words from documents
from nltk.stem import WordNetLemmatizer #Used to lemmatize words
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import Normalizer
from sklearn import metrics

from sklearn.cluster import KMeans

import sys
from time import time

import pandas as pd
import numpy as np
```

[2]:
```
# Selected 3 categories from the 20 newsgroups dataset

categories = [
    'talk.religion.misc',
    'comp.graphics',
    'sci.space',
]

print("Loading 20 newsgroups dataset for categories:")
print(categories)
```

```
Loading 20 newsgroups dataset for categories:
['talk.religion.misc', 'comp.graphics', 'sci.space']
```

[3]:
```
df = fetch_20newsgroups(subset='all', categories=categories,
                             shuffle=False, remove=('headers', 'footers',␣
 ↪'quotes'))
```

[4]:
```
labels = df.target
true_k = len(np.unique(labels)) ## This should be 3 in this example
print(true_k)
```

```
3
```

[5]:

```
#Perform Lemmatization
```

[6]:

```
lemmatizer = WordNetLemmatizer()
for i in range(len(df.data)):
    word_list = word_tokenize(df.data[i])
    lemmatized_doc = ""
    for word in word_list:
        lemmatized_doc = lemmatized_doc + " " + lemmatizer.lemmatize(word)
    df.data[i] = lemmatized_doc
```

[7]:

```
print(df.data[1])
```

```
 In regard to fractal commpression , I have seen 2 fractal compressed `` movie
'' . They were both fairly impressive . The first one wa a 64 gray scale ``
movie '' of Casablanca , it wa 1.3MB and had 11 minute of 13 fps video . It wa a
little grainy but not bad at all . The second one I saw wa only 3 minute but it
had 8 bit color with 10fps and measured in at 1.2MB . I consider the fractal
movie a practical thing to explore . But unlike many other format out there ,
you do end up losing resolution . I do n't know what kind of software/hardware
wa used for creating the `` movie '' I saw but the guy that showed them to me
said it took 5-15 minute per frame to generate . But a I said above playback wa
10 or more frame per second . And how else could you put 11 minute on one floppy
disk ?
```

[8]:

```
#We next convert our corpus into tf-idf vectors. We remove common stop words,␣
 ↪terms with very low document frequency (many of them are numbers
#or misspells), accents.
```

[9]:

```
vectorizer = TfidfVectorizer(strip_accents='unicode', stop_words='english',␣
 ↪min_df=2) ## Corpus is in English
X = vectorizer.fit_transform(df.data)
```

[10]:

```
print(X.shape)
```

```
(2588, 14439)
```

[11]:

```
#Clustering using standard k-means¶
```

[12]:

```
#We choose a value that reflects our knowledge about the data
```

[13]:

```
#We may try several value, possibly in increasing order
```

[14]:

```
#we set  k=3
```

[15]:

```
km = KMeans(n_clusters=true_k, init='k-means++', max_iter=100)
t0 = time()
km.fit(X)
print("done in %0.3fs" % (time() - t0))
```

```
done in 0.162s
```

```
/home/nmit/anaconda3/lib/python3.11/site-
packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  warnings.warn(
```

[16]:

```
#Standard measures of cluster quality¶
```

[17]:

```
print("Homogeneity: %0.3f" % metrics.homogeneity_score(labels, km.labels_))
print("Completeness: %0.3f" % metrics.completeness_score(labels, km.labels_))
print("V-measure: %0.3f" % metrics.v_measure_score(labels, km.labels_))
print("Adjusted Rand-Index: %.3f"
      % metrics.adjusted_rand_score(labels, km.labels_))
print("Silhouette Coefficient: %0.3f"
      % metrics.silhouette_score(X, km.labels_, sample_size=1000))
```

```
Homogeneity: 0.293
Completeness: 0.401
V-measure: 0.339
Adjusted Rand-Index: 0.210
Silhouette Coefficient: 0.009
```

[18]:

```
#Identify the 10 most relevant terms in each cluster¶
```

[20]:

```
centroids = km.cluster_centers_.argsort()[:, ::-1] ## Indices of largest␣
 ↪centroids' entries in descending order
#vectorizer.get_feature_names use instead vectorizer.get_feature_names_out
terms = vectorizer.get_feature_names_out()
for i in range(true_k):
    print("Cluster %d:" % i, end='')
    for ind in centroids[i, :10]:
        print(' %s' % terms[ind], end='')
    print()
```

```
Cluster 0: god jesus christian bible wa people christians say christ did
Cluster 1: wa space just like think ha time know did people
Cluster 2: file image thanks format program know graphic ftp bit gif
```

[21]:

```
#Visualization
```

[22]:

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

[23]:

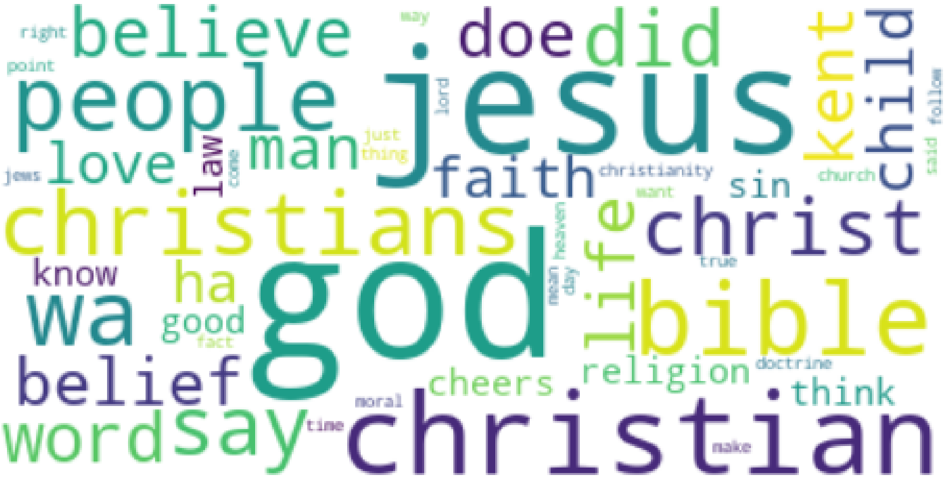
```
def frequencies_dict(cluster_index):
    if cluster_index > true_k - 1:
        return
    term_frequencies = km.cluster_centers_[cluster_index]
    sorted_terms = centroids[cluster_index]
    frequencies = {terms[i]: term_frequencies[i] for i in sorted_terms}
    return frequencies
```

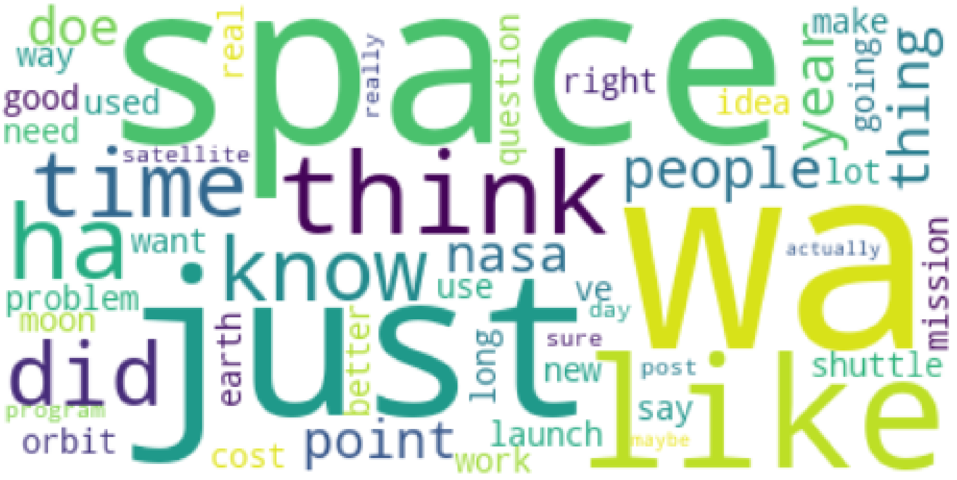
[24]:

```
def makeImage(frequencies):

    wc = WordCloud(background_color="white", max_words=50)
    # generate word cloud
    wc.generate_from_frequencies(frequencies)

    # show
    plt.imshow(wc, interpolation="bilinear")
    plt.axis("off")
    plt.show()
```

[25]:

```
for i in range(true_k):
    freq = frequencies_dict(i)
    makeImage(freq)
    print()
```




space
wa
just
like
think
time
ha
know
did
people
thing
year
nasa
point
doe
way
good
used
need
real
really
question
right
idea
make
going
lot
satellite
want
problem
moon
earth
better
use
long
ve
day
sure
new
post
shuttle
mission
actually
program
orbit
cost
work
launch
say
maybe

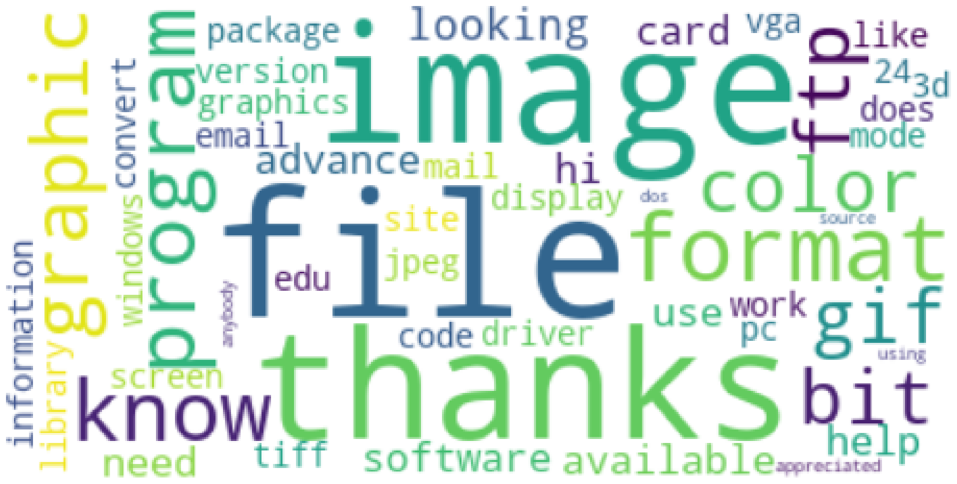


image
file
thanks
format
graphic
program
know
color
gif
bit
ftp
package
looking
card
vga
like
24
3d
does
mode
version
graphics
email
convert
advance
mail
hi
display
dos
source
site
jpeg
edu
windows
information
library
anybody
code
driver
use
work
pc
using
screen
need
tiff
software
available
help
appreciated

[ ]: