Assignment Project Report

PLSA: Text Document Clustering

Name : Meghna Maan

Course : AI & ML (Batch 4)

Given Question

Perform topic modelling using the 20 Newsgroup dataset (the dataset is also available in sklearn datasets sub-module). Perform the required data cleaning steps using NLP and then model the topics

1. Using Latent Dirichlet Allocation (LDA).

2. Using Probabilistic Latent Semantic Analysis (PLSA)

Prerequisites

1. Software:

   Python 3

2. Tools:

   - Numpy
   - Pandas
   - Matplotlib
   - Nltk
   - Sklearn

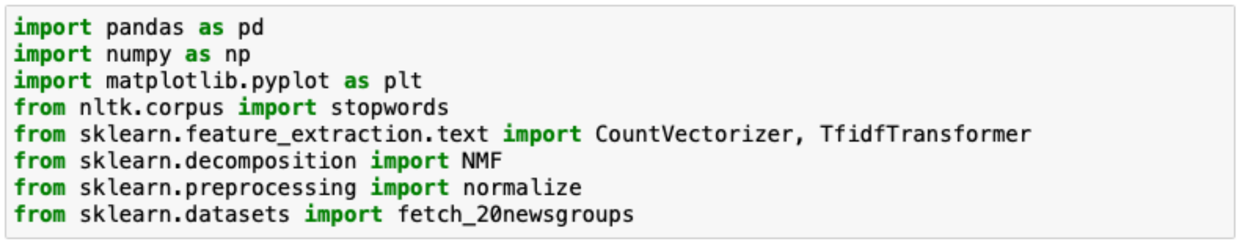
Dataset: https://scikit-learn.org/0.19/datasets/twenty_newsgroups.html

Methods Used

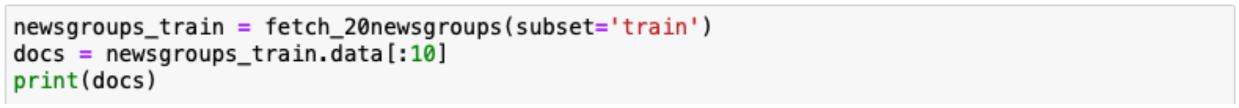
PLSA or Probabilistic Latent Semantic Analysis is a technique used to model information under a probabilistic framework. It is a statistical technique for the analysis of two-mode and co-occurrence data. PLSA characterizes each word in a document as a sample from a mixture model, where mixture components are conditionally independent multinomial distributions. Its main goal is to model cooccurrence information under a probabilistic framework in order to discover the underlying semantic structure of the data.

Implementation

1. Loading Libraries and Dataset

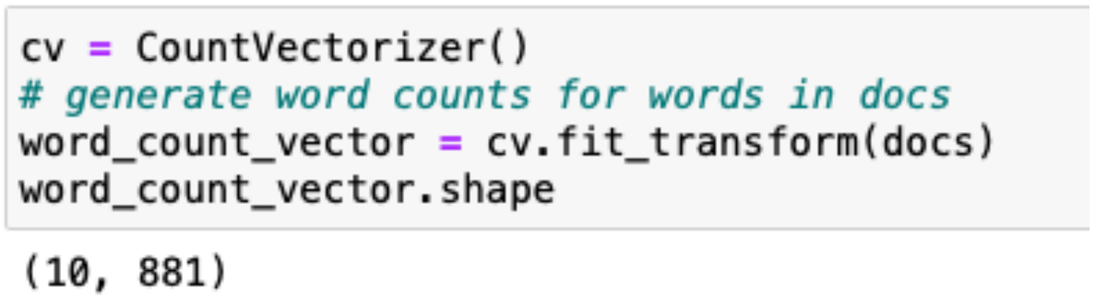


```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.decomposition import NMF
from sklearn.preprocessing import normalize
from sklearn.datasets import fetch_20newsgroups
```



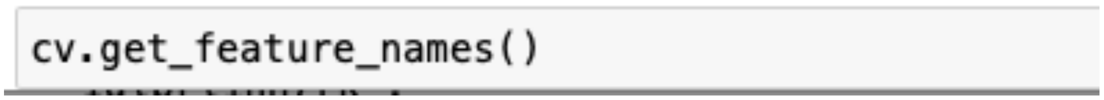
```
newsgroups_train = fetch_20newsgroups(subset='train')
docs = newsgroups_train.data[:10]
print(docs)
```

2. Generating the word count and printing the feature names



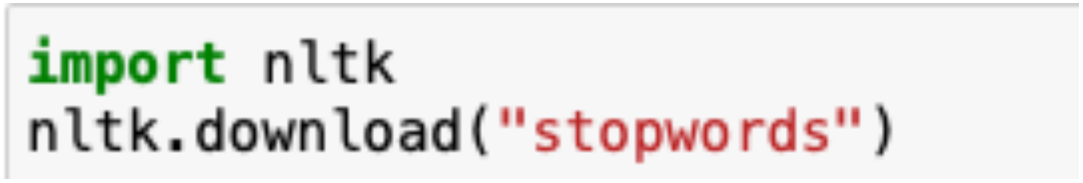
```
cv = CountVectorizer()
# generate word counts for words in docs
word_count_vector = cv.fit_transform(docs)
word_count_vector.shape
```

```
(10, 881)
```

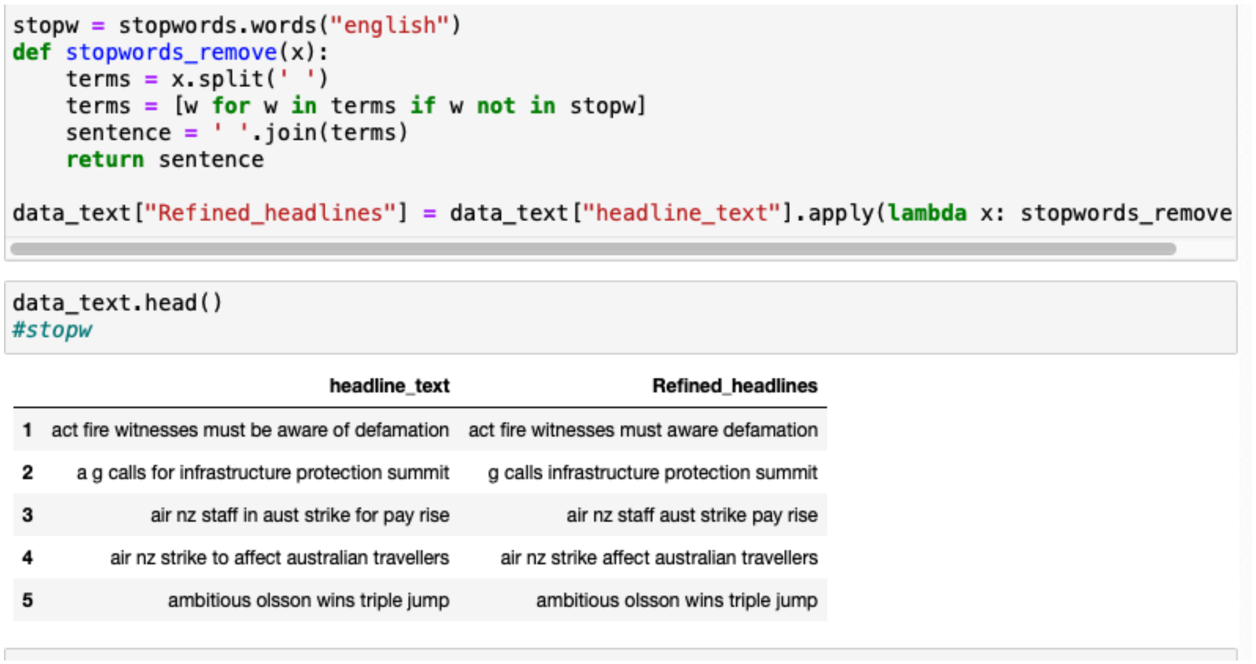


```
cv.get_feature_names()
```

3. importing nltk and checking the words along with priting of data



```
import nltk
nltk.download("stopwords")
```



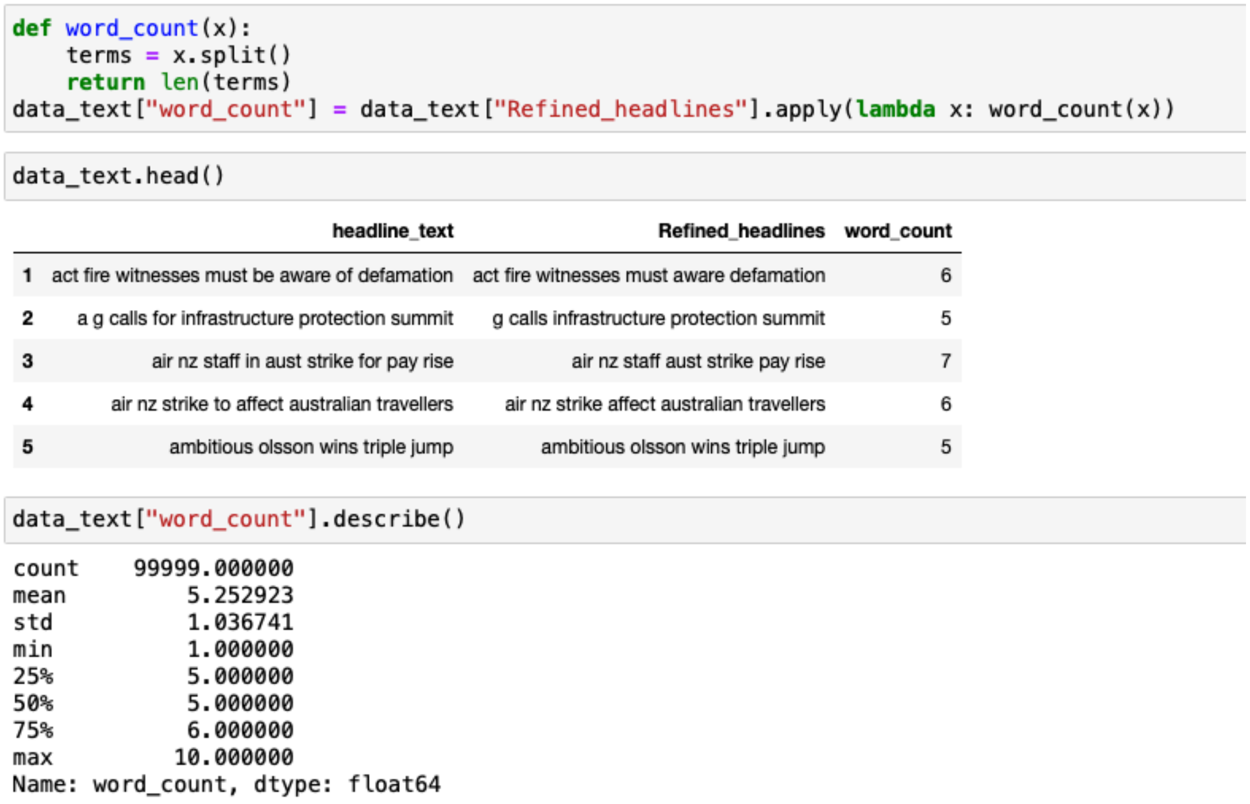
```
stopw = stopwords.words("english")
def stopwords_remove(x):
    terms = x.split(' ')
    terms = [w for w in terms if w not in stopw]
    sentence = ' '.join(terms)
    return sentence

data_text["Refined_headlines"] = data_text["headline_text"].apply(lambda x: stopwords_remove
```

```
data_text.head()
#stopw
```

	headline_text	Refined_headlines
1	act fire witnesses must be aware of defamation	act fire witnesses must aware defamation
2	a g calls for infrastructure protection summit	g calls infrastructure protection summit
3	air nz staff in aust strike for pay rise	air nz staff aust strike pay rise
4	air nz strike to affect australian travellers	air nz strike affect australian travellers
5	ambitious olsson wins triple jump	ambitious olsson wins triple jump

4. Getting the word count



```
def word_count(x):
    terms = x.split()
    return len(terms)
data_text["word_count"] = data_text["Refined_headlines"].apply(lambda x: word_count(x))
```

```
data_text.head()
```

	headline_text	Refined_headlines	word_count
1	act fire witnesses must be aware of defamation	act fire witnesses must aware defamation	6
2	a g calls for infrastructure protection summit	g calls infrastructure protection summit	5
3	air nz staff in aust strike for pay rise	air nz staff aust strike pay rise	7
4	air nz strike to affect australian travellers	air nz strike affect australian travellers	6
5	ambitious olsson wins triple jump	ambitious olsson wins triple jump	5

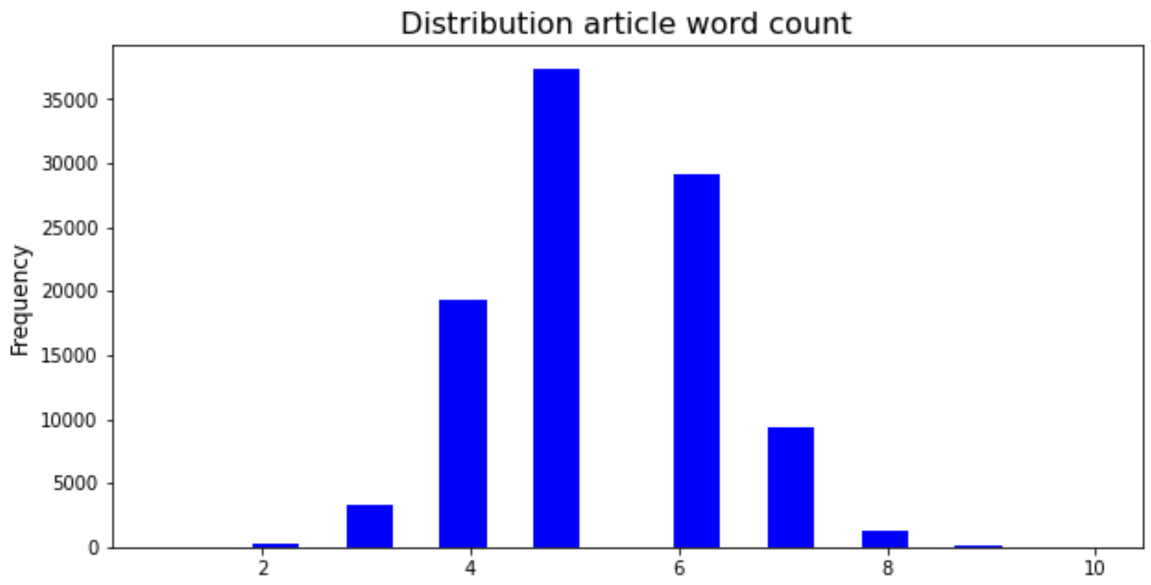
```
data_text["word_count"].describe()
```

```
count    99999.000000
mean         5.252923
std          1.036741
min          1.000000
25%          5.000000
50%          5.000000
75%          6.000000
max         10.000000
Name: word_count, dtype: float64
```

5. Plotting Graph for word count distribution

```
fig = plt.figure(figsize = (10, 5))
plt.hist(data_text["word_count"], bins = 20, color = "blue")
plt.title("Distribution article word count", fontsize = 16)
plt.ylabel("Frequency", fontsize = 12)
plt.xlabel("word_count", fontsize = 12)
```

```
Text(0.5, 0, 'word_count')
```



6. Final Result

```
num_topics = 5

model = NMF(n_components = num_topics, init = "nndsvd")
model.fit(x_tfidf_norm)

def get_nmf_topics(model, n_top_words):
    feat_names = vectorizer.get_feature_names()

    word_dict = {}
    for i in range(num_topics):
        words_ids = model.components_[i].argsort()[:-n_top_words-1:-1]
        words = [feat_names[key] for key in words_ids]

        word_dict['Topic #' + '{:02d}'.format(i+1)] = words
    return pd.DataFrame(word_dict)
```

```
get_nmf_topics(model, 30)
```