VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY

Department of Computer Engineering

Mini-Project Report on

Document Clustering

NLP

Submitted by
Aniket Bote (D17C-10)
Devashish Gopalani (D17C-20)
Ashish Gwalani (D17C-22)

INDEX :-

1.INTRODUCTION

1.1 PROBLEM DEFINITION:

Clustering is an automatic learning technique aimed at grouping a set of objects into subsets or clusters. The goal is to create clusters that are coherent internally, but substantially different from each other. In plain words, objects in the same cluster should be as similar as possible, whereas objects in one cluster should be as dissimilar as possible from objects in the other clusters

We can define the goal in hard flat clustering as follows. Given (i) a set of documents D = {d1,...dn}, (ii) a desired number of clusters K, and (iii) an *objective function* that evaluates the quality of a clustering, we want to compute an assignment Υ:D$\rightarrow$ {1,...,K} that minimizes (or, in other cases, maximizes) the objective function. In most cases, we also demand that Υ is surjective, i.e., that none of the K clusters is empty.

1.2 SCOPE OF THE PROJECT:

Most of the MNCs have their documents in digital format. If they are not arranged and sorted properly the company may face huge problems. The problems they face may be as small as delay in their working to loses in profit due improper utilization of available information.

The project deals with clustering of documents into specified clusters. The project does not need to know the number of clusters in advance. The number of clusters is calculated by system itself using the Elbow Curve method.

Apart from MNCs, the project is also helpful for advertisement sector. With the help of these cluster, we can analyze if people are responsive only to advertisement from particular cluster.

1.3 REQUIREMENTS & MINIMUM FUNCTIONALITY:

- Segregated document cluster should be generated as the end result.
- User should have the choice of selecting the number of clusters.
- System requires GPU for model training.

1.4 TECHNOLOGIES TO BE USED:

1. Python 3.6

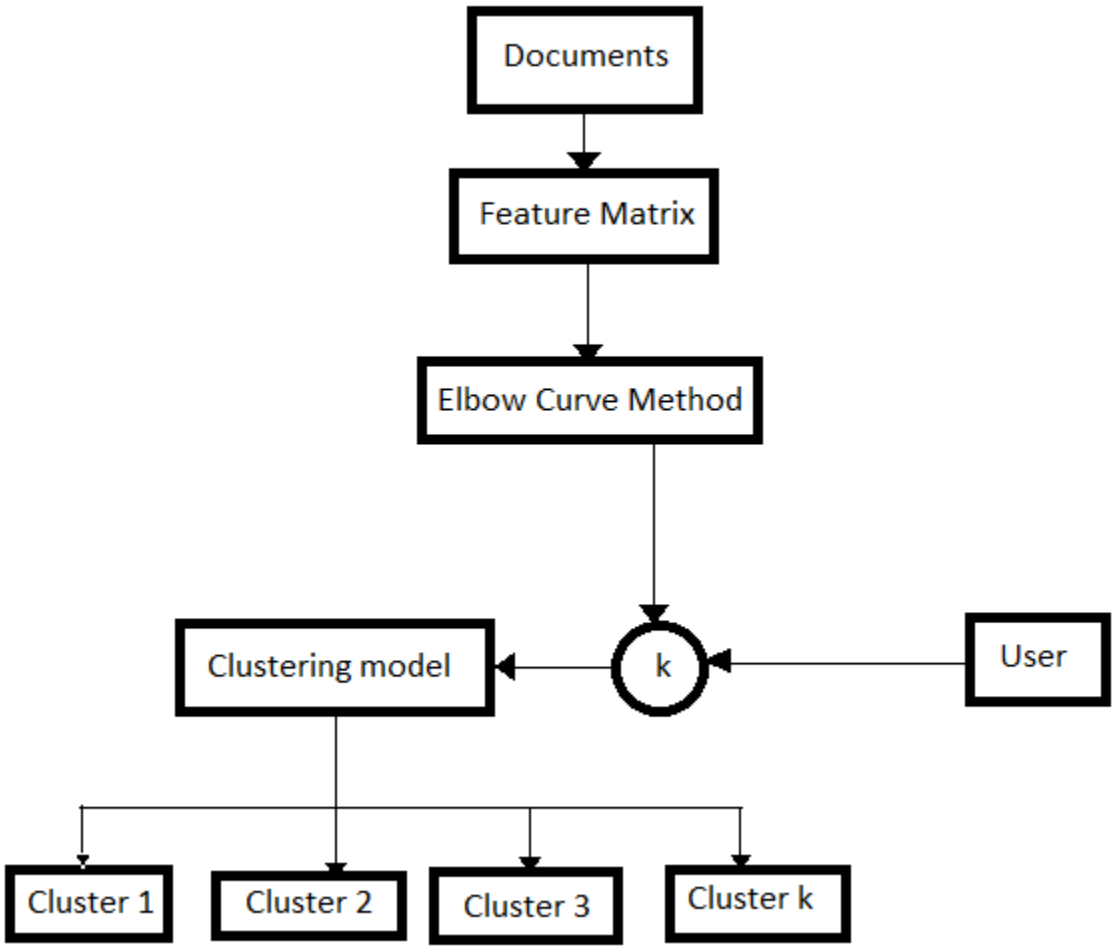
2. NLTK

3. Scikit-Learn

4. Matplotlib

5. regex

3.CONCEPTUAL SYSTEM DIAGRAM:



4.IMPLEMENTATION:

CODE :

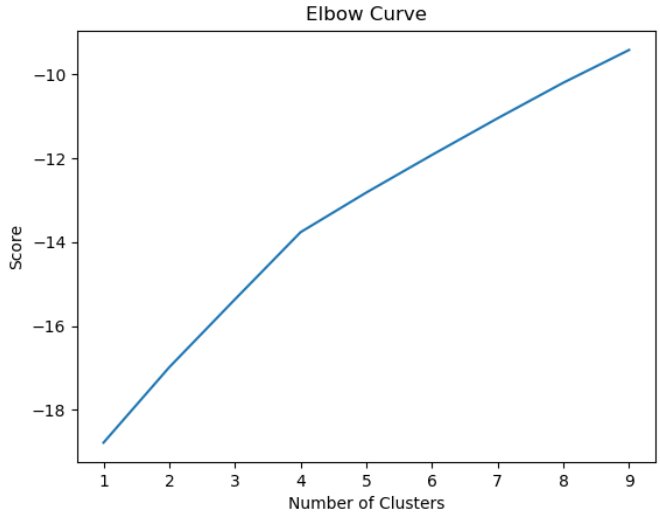
```
"""### Import Statements"""

import os
import random
import nltk
import re
from nltk.stem.snowball import
SnowballStemmer
from sklearn.feature_extraction.text import
TfidfVectorizer
from sklearn.cluster import KMeans
from sklearn import metrics
import matplotlib.pyplot as plt


"""### Downloading extra dependencies
from NLTK"""

# nltk.download('punkt')
# nltk.download('stopwords')


"""### Getting stopwords customized to
your problem statement"""

#Use this function to create custom list of
stop_words for your Project

path =
r'C:\Users\Aniket\Desktop\Experiments\NL
P\Document-Clustering-TFIDF\Git
Clustering\Stopwords\stopwords_not_to_be
_used.txt' #Add the path to
stopwords_not_to_be_used.txt file
def get_stopwords(path):
  stopwords =
nltk.corpus.stopwords.words('english')
  not_words = []
  with open(path,'r') as f:
    not_words.append(f.readlines())
  not_words = [word.replace('\n','') for words
in not_words for word in words]
  not_words = set(not_words)
  stopwords = set(stopwords)
  customized_stopwords = list(stopwords -
not_words)
  return stopwords,customized_stopwords

stop_words,customized_stopwords =
get_stopwords(path)


"""### Loading the Data"""

path =
r'C:\Users\Aniket\Desktop\Experiments\NL
P\Document-Clustering-TFIDF\Git
Clustering\Articles' #Add the path to
Articles folder
seed = 137 #Seed value
def load_data(path,seed):
  train_texts = []
  for fname in sorted(os.listdir(path)):
    if fname.endswith('.txt'):
      with open(os.path.join(path,fname),'r') as
f:
        train_texts.append(f.read())
  random.seed(seed)
  random.shuffle(train_texts)
  return train_texts
```

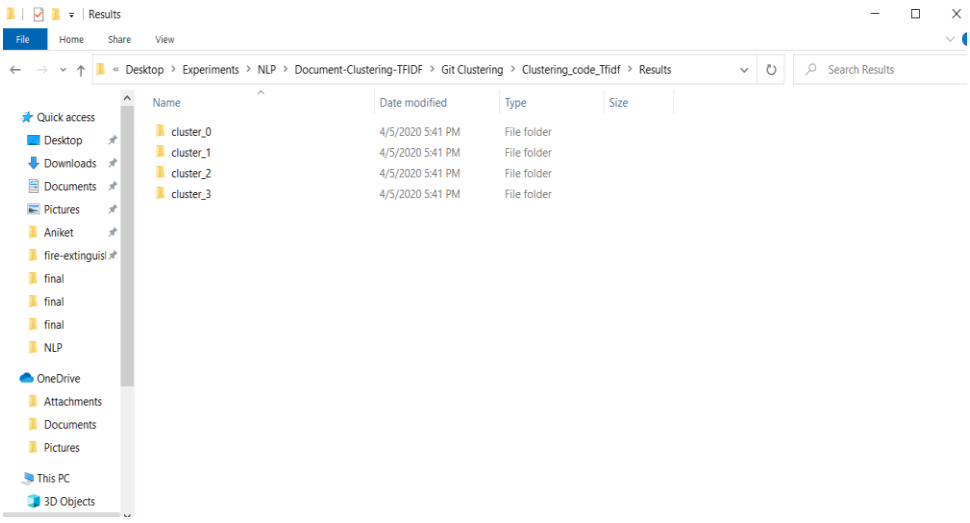
```
train_texts = load_data(path,seed)


"""### Tokenizing the document and
filtering the tokens"""

def tokenize(train_texts):
  filtered_tokens = []
  tokens = [word for sent in
nltk.sent_tokenize(train_texts) for word in
nltk.word_tokenize(sent)]
  for token in tokens:
    if re.search('[a-zA-Z]',token):
      filtered_tokens.append(token)
  return filtered_tokens


"""### Tokenizing and stemming using
Snowball stemmer"""

def tokenize_stem(train_texts):
  tokens = tokenize(train_texts)
  stemmer = SnowballStemmer('english')
  stemmed_tokens = [stemmer.stem(token)
for token in tokens]
  return stemmed_tokens



"""### Generating the vocab for problem
statement"""

def generate_vocab(train_texts):
  vocab_tokenized = []
  vocab_stemmed = []
  total_words = []
  for text in train_texts:
    allwords_tokenized = tokenize(text)
    total_words.append(allwords_tokenized)
vocab_tokenized.extend(allwords_tokenized
)
    allwords_stemmed = tokenize_stem(text)

vocab_stemmed.extend(allwords_stemmed)
  return
vocab_tokenized,vocab_stemmed,total_wor
ds
vocab_tokenized,vocab_stemmed,total_wor
ds = generate_vocab(train_texts)


"""### Calculating Tf-idf matrix"""

'''
Attributes in TfidVectorizer are data
dependent.
Use 'stop_words = customized_stopwords' if
you want to use your own set of stopwords
else leave it as it is.
Functions available for tokenizer ->
1)tokenize  2) tokenize_stem  3) Remove the
attribute to use default function
'''

def tfid_vector(train_texts):
  tfidf_vectorizer = TfidfVectorizer(max_df
= 0.85, min_df = 0.1, sublinear_tf = True,
stop_words = 'english', use_idf = True,
tokenizer = tokenize, ngram_range = (1,10))
  tfidf_matrix =
tfidf_vectorizer.fit_transform(train_texts)
  return tfidf_matrix
tfidf_matrix = tfid_vector(train_texts)


"""### Clustering Using K - Means"""

#Code For Elbow Method
```

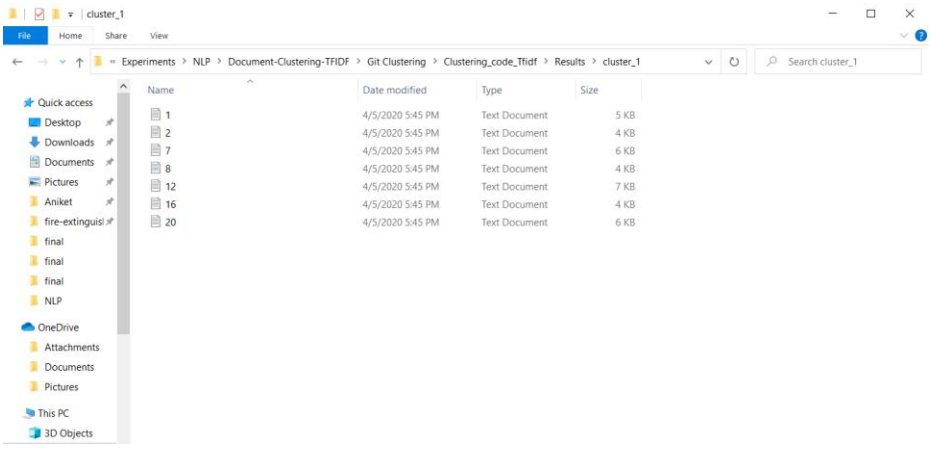
```
nc = range(1,10)
kmeans = [KMeans(n_clusters = i, n_init =
100, max_iter = 500, precompute_distances
= 'auto' ) for i in nc]
score =
[kmeans[i].fit(tfidf_matrix).score(tfidf_matri
x) for i in range(len(kmeans))]
plt.plot(nc,score)
plt.xlabel('Number of Clusters')
plt.ylabel('Score')
plt.title('Elbow Curve')
plt.show()


#Uncomment the below code after getting
appropriate k value from the graph

K_value = int(input("Enter Optimum K
Value = "))      #Write the optimum K-value
after seeing the Elbow Graph

km = KMeans(n_clusters = K_value, n_init
= 2000, max_iter = 6000,
precompute_distances = 'auto' )
clusters = km.fit_predict(tfidf_matrix)
clusters = list(clusters)

if 'Results' not in os.listdir(os.getcwd()):
    os.mkdir('Results')

F_count = 0
for c,t in zip(clusters, train_texts):
    if 'cluster_{}'.format(c) not in
os.listdir('Results'):
        os.mkdir('Results/cluster_{}'.format(c))
    f =
open('Results/cluster_{}/{}.txt'.format(c,F_c
ount),'w')
    f.write(t)
    f.close()
    F_count += 1
```

Execution :

```
C:\Users\Aniket\Desktop\Experiments\NLP\Document-Clustering-TFIDF\Git Clusteri
ng\Clustering_code_Tfidf>python clustering_Documents.py
Enter Optimum K Value = 4
[1, 3, 3, 0, 2, 2, 2, 3, 3, 1, 0, 1, 3, 0, 2, 1, 3, 1, 1, 1, 3, 0, 2]

C:\Users\Aniket\Desktop\Experiments\NLP\Document-Clustering-TFIDF\Git Clusteri
ng\Clustering_code_Tfidf>
```


Elbow Curve
Score
−10
−12
−14
−16
−18
1
2
3
4
5
6
7
8
9
Number of Clusters


Results
File
Home
Share
View
Desktop › Experiments › NLP › Document-Clustering-TFIDF › Git Clustering › Clustering_code_Tfidf › Results
Search Results
Quick access
Desktop
Downloads
Documents
Pictures
Aniket
fire-extinguisl
final
final
final
NLP
OneDrive
Attachments
Documents
Pictures
This PC
3D Objects
Name
Date modified
Type
Size
cluster_0 4/5/2020 5:41 PM File folder
cluster_1 4/5/2020 5:41 PM File folder
cluster_2 4/5/2020 5:41 PM File folder
cluster_3 4/5/2020 5:41 PM File folder


cluster_1
File
Home
Share
View
Experiments › NLP › Document-Clustering-TFIDF › Git Clustering › Clustering_code_Tfidf › Results › cluster_1
Search cluster_1
Quick access
Desktop
Downloads
Documents
Pictures
Aniket
fire-extinguisl
final
final
final
NLP
OneDrive
Attachments
Documents
Pictures
This PC
3D Objects
Name
Date modified
Type
Size
1 4/5/2020 5:45 PM Text Document 5 KB
2 4/5/2020 5:45 PM Text Document 4 KB
7 4/5/2020 5:45 PM Text Document 6 KB
8 4/5/2020 5:45 PM Text Document 4 KB
12 4/5/2020 5:45 PM Text Document 7 KB
16 4/5/2020 5:45 PM Text Document 4 KB
20 4/5/2020 5:45 PM Text Document 6 KB

Conclusion:

Document clustering (or text clustering) is the application of cluster analysis to textual documents. It has applications in automatic document organization, topic extraction and fast information retrieval or filtering.