Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA: 3.18)

JUNAID GIRKAR | 60004190057 | BE COMPS A2 | WEB INTELLIGENCE

EXPERIMENT - 4

AIM: Latent Semantic Indexing

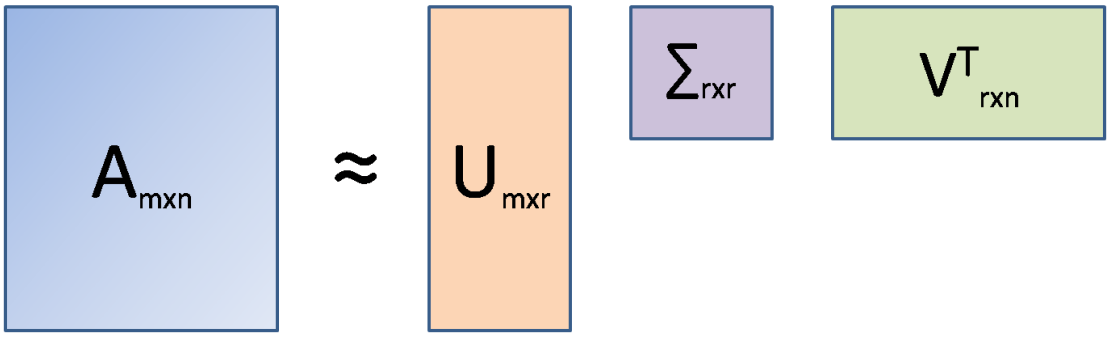
THEORY:
Latent Semantic Analysis (LSA) involves creating structured data from a collection of unstructured texts. Before getting into the concept of LSA, let us have a quick intuitive understanding of the concept. When we write anything like text, the words are not chosen randomly from a vocabulary.

Rather, we think about a theme (or topic) and then chose words such that we can express our thoughts to others in a more meaningful way. This theme or topic is usually considered as a latent dimension.

It is latent because we can't see the dimension explicitly. Rather, we understand it only after going through the text. This means that most of the words are semantically linked to other words to express a theme. So, if words are occurring in a collection of documents with varying frequencies, it should indicate how different people try to express themselves using different words and different topics or themes.

In other words, word frequencies in different documents play a key role in extracting the latent topics. LSA tries to extract the dimensions using a machine learning algorithm called Singular Value Decomposition or SVD.

Singular Value Decomposition or SVD is essentially a matrix factorization technique. In this method, any matrix can be decomposed into three parts as shown below.



Here, A is the document-term matrix (documents in the rows(m), unique words in the columns(n), and frequencies at the intersections of documents and words). It is to be kept in mind that in LSA, the original document-term matrix is approximated by way of multiplying three other matrices, i.e., U, ∑ and VT. Here, r is the number of aspects or topics. Once we fix r (r<<n) and run SVD, the outcome that comes out is called Truncated SVD and LSA is essentially a truncated SVD only.

SVD is used in such situations because, unlike PCA, SVD does not require a correlation matrix or a covariance matrix to decompose. In that sense, SVD is free from any normality assumption of data (covariance calculation assumes a normal distribution of data). The U matrix is the document-aspect matrix, V is the word-aspect matrix, and ∑ is the diagonal matrix of the singular values. Similar to PCA, SVD also combines columns of the original matrix linearly to arrive at the U matrix. To arrive at the V matrix, SVD combines the rows of the original matrix linearly. Thus, from a sparse document-term matrix, it is possible to get a dense document-aspect matrix that can be used for either document clustering or document classification using available ML tools. The V matrix, on the other hand, is the word embedding matrix (i.e. each and every word is expressed by r floating-point numbers) and this matrix can be used in other sequential modeling tasks.

CODE:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
pd.set_option("display.max_colwidth", 200)

from sklearn.datasets import fetch_20newsgroups
dataset = fetch_20newsgroups(shuffle=True, random_state=1, remove=('headers',
'footers', 'quotes'))
documents = dataset.data
Dataset.target_names
```

OUTPUT

```
['alt.atheism',
 'comp.graphics',
```

```
'comp.os.ms-windows.misc',
'comp.sys.ibm.pc.hardware',
'comp.sys.mac.hardware',
'comp.windows.x',
'misc.forsale',
'rec.autos',
'rec.motorcycles',
'rec.sport.baseball',
'rec.sport.hockey',
'sci.crypt',
'sci.electronics',
'sci.med',
'sci.space',
'soc.religion.christian',
'talk.politics.guns',
'talk.politics.mideast',
'talk.politics.misc',
'talk.religion.misc']
```

CODE:

```
news_df = pd.DataFrame({'document':documents})
# remove everything except alphabets`
news_df['clean_doc'] = news_df['document'].str.replace("[^a-zA-Z]", " ")
# remove short words
news_df['clean_doc']=news_df['clean_doc'].apply(lambda x:' '.join([w for w in
x.split() if len(w)>3]))
# make all text lowercase
news_df['clean_doc'] = news_df['clean_doc'].apply(lambda x: x.lower())

from nltk.corpus import stopwords
import nltk
nltk.download('stopwords')
stopwords = nltk.corpus.stopwords.words('english')
# stop_words = stopwords.words('english')
# tokenization
tokenized_doc = news_df['clean_doc'].apply(lambda x: x.split())
```

```
# remove stop-words
tokenized_doc = tokenized_doc.apply(lambda x: [item for item in x if item not in
stopwords])
# de-tokenization
detokenized_doc = []
for i in range(len(news_df)):
    t = ' '.join(tokenized_doc[i])
    detokenized_doc.append(t)
news_df['clean_doc'] = detokenized_doc


from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(stop_words='english', max_features= 1000, max_df =
0.5, smooth_idf=True)
X = vectorizer.fit_transform(news_df['clean_doc'])
X.shape


from sklearn.decomposition import TruncatedSVD
# SVD represent documents and terms in vectors
svd_model = TruncatedSVD(n_components=20, algorithm='randomized',
n_iter=100, random_state=122)
svd_model.fit(X)


terms = vectorizer.get_feature_names_out()
for i, comp in enumerate(svd_model.components_):
    terms_comp = zip(terms, comp)
    sorted_terms = sorted(terms_comp, key= lambda x:x[1],
    reverse=True)[:7]
    print("Topic "+str(i)+": ", end="")
    for t in sorted_terms:
        print(t[0], end="")
        print(" ", end="\t")
    print()
```

JUNAID GIRKAR | 60004190057 | BE COMPS A2 | WEB INTELLIGENCE

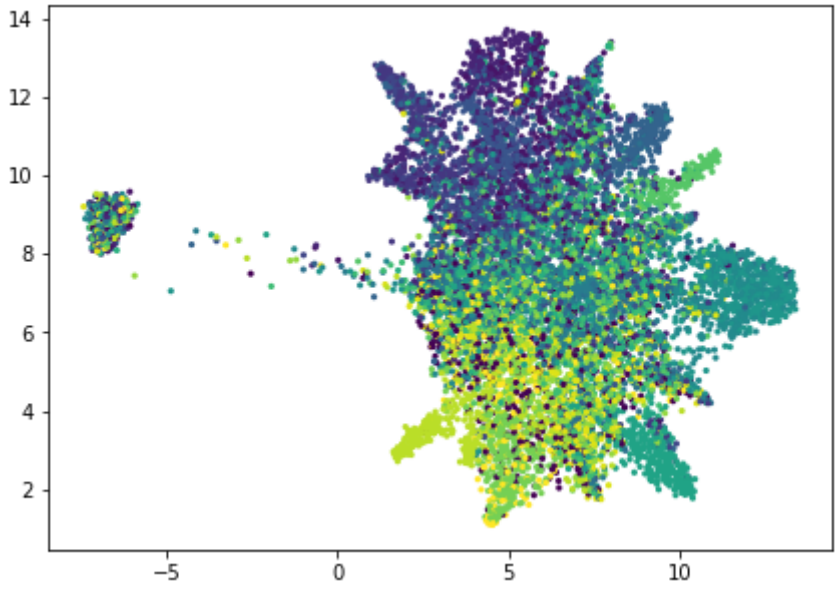
OUTPUT:

```
Topic 0: like know people think good time thanks
Topic 1: thanks windows card drive mail file advance
Topic 2: game team year games season players good
Topic 3: drive scsi hard disk card drives problem
Topic 4: windows file window files program using problem
Topic 5: chip government mail space information encryption data
Topic 6: like bike chip know sounds looks look
Topic 7: card video sale monitor offer price jesus
Topic 8: know card chip government video people clipper
Topic 9: good know time bike jesus problem work
Topic 10: think chip good thanks clipper encryption need
Topic 11: thanks good right bike problem people time
Topic 12: good people windows know file sale files
Topic 13: space think know nasa problem year israel
Topic 14: space good card people time nasa thanks
Topic 15: people problem window time game want work
Topic 16: time bike right windows file need really
Topic 17: time problem file think israel long mail
Topic 18: file need card files right problem good
Topic 19: problem file thanks used space chip sale
```

CODE:

```
import umap.umap_ as umap
X_topics = svd_model.fit_transform(X)
embedding = umap.UMAP(n_neighbors=150, min_dist=0.5,
random_state=12).fit_transform(X_topics)
plt.figure(figsize=(7,5))
plt.scatter(embedding[:, 0], embedding[:, 1],
c = dataset.target,
s = 10, # size
edgecolor='none' )
plt.show()
```

OUTPUT:



CONCLUSION: Understanding the context behind a sentence is an important part behind understanding its meaning. This comes naturally to the human brain, but is difficult for the computer to understand. Latent Semantic Analysis (LSA) is what comes in place to help the computer understand the context while during NLP. In this experiment, we have implemented LSA using python.