Project Report

PLSA: Text Document Clustering

NAME: HARI KRISHNAN

COURSE: AI and ML

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

What things you need to install the software and how to install them:

Python 3.6 This setup requires that your machine has latest version of python. The following url https://www.python.org/downloads/ can be referred to download python. Once you have python downloaded and installed, you will need to setup PATH variables (if you want to run python program directly, detail instructions are below in how to run software section). To do that check this: https://www.pythoncentral.io/add-python-to-path-python-is-not-recognized-asan-internal-or-externalcommand/ . Setting up PATH variable is optional as you can also run program without it and more instruction are given below on this topic.

Second and easier option is to download anaconda and use its anaconda prompt to run the commands. To install anaconda check this url https://www.anaconda.com/download/ You will also need to download and install below 3 packages after you install either python or anaconda from the steps above Sklearn (scikit-learn) numpy scipy if you have chosen to install python 3.6

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

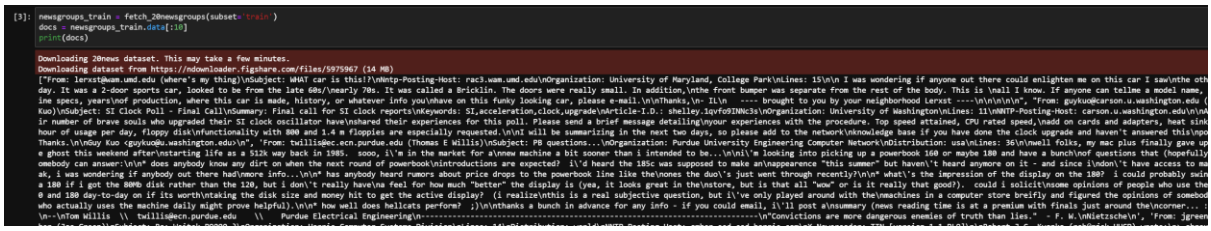
Load all required libraries and Dateset

```
[1]: ! pip install nltk

Requirement already satisfied: nltk in c:\users\dhiva\anaconda3\lib\site-packages (3.4.5)
Requirement already satisfied: six in c:\users\dhiva\anaconda3\lib\site-packages (from nltk) (1.15.0)

[2]: 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
```

Downloading dataset



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

Downloading 20news dataset. This may take a few minutes.
Downloading dataset from https://ndownloader.figshare.com/files/5975967 (14 MB)
["From: lerxst@wam.umd.edu (where's my thing)\nSubject: WHAT car is this!?\nNntp-Posting-Host: rac3.wam.umd.edu\nOrganization: University of Maryland, College Park\nLines: 15\n\n I was wondering if anyone out there could enlighten me on this car I saw\nthe oth
day. It was a 2-door sports car, looked to be from the late 60s/\nearly 70s. It was called a Bricklin. The doors were really small. In addition,\nthe front bumper was separate from the rest of the body. This is \nall I know. If anyone can tellme a model name,
ine specs, years\nof production, where this car is made, history, or whatever info you\nhave on this funky looking car, please e-mail.\n\nThanks,\n- IL\n   ---- brought to you by your neighborhood Lerxst ----\n\n\n\n\n", "From: guykuo@carson.u.washington.edu (
Kuo)\nSubject: SI Clock Poll - Final Call\nSummary: Final call for SI clock reports\nKeywords: SI,acceleration,clock,upgrade\nArticle-I.D.: shelley.1qvfo9INNc3s\nOrganization: University of Washington\nLines: 11\nNNTP-Posting-Host: carson.u.washington.edu\n\nA
ir number of brave souls who upgraded their SI clock oscillator have\nshared their experiences for this poll. Please send a brief message detailing\nyour experiences with the procedure. Top speed attained, CPU rated speed,\nadd on cards and adapters, heat sink
hour of usage per day, floppy disk\nfunctionality with 800 and 1.4 m floppies are especially requested.\n\nI will be summarizing in the next two days, so please add to the network\nknowledge base if you have done the clock upgrade and haven't answered this\npo
Thanks.\n\nGuy Kuo <guykuo@u.washington.edu>\n", "From: twillis@ec.ecn.purdue.edu (Thomas E Willis)\nSubject: PB questions...\nOrganization: Purdue University Engineering Computer Network\nDistribution: usa\nLines: 36\n\nwell folks, my mac plus finally gave up
e ghost this weekend after\nstarting life as a 512k way back in 1985.  sooo, i\'m in the market for a\nnew machine a bit sooner than i intended to be...\n\ni\'m looking into picking up a powerbook 160 or maybe 180 and have a bunch\nof questions that (hopefully
omebody can answer:\n\n* does anybody know any dirt on when the next round of powerbook\nintroductions are expected?  i\'d heard the 185c was supposed to make an\nappearence "this summer" but haven\'t heard anymore on it - and since i\ndon\'t have access to ma
ak, i was wondering if anybody out there has\nmore info...\n\n* has anybody heard rumors about price drops to the powerbook line like the\nones the duo\'s just went through recently?\n\n* what\'s the impression of the display on the 180?  i could probably swin
a 180 if i got the 80Mb disk rather than the 120, but i don\'t really have\na feel for how much "better" the display is (yea, it looks great in the\nstore, but is that all "wow" or is it really that good?).  could i solicit\nsome opinions of people who use the
0 and 180 day-to-day on if its worth\ntaking the disk size and money hit to get the active display?  (i realize\nthis is a real subjective question, but i\'ve only played around with the\nmachines in a computer store breifly and figured the opinions of somebod
who actually uses the machine daily might prove helpful).\n\n* how well does hellcats perform?  ;)\n\nthanks a bunch in advance for any info - if you could email, i\'ll post a\nsummary (news reading time is at a premium with finals just around the\ncorner... :
\n--\nTom Willis  \\  twillis@ecn.purdue.edu    \\    Purdue Electrical Engineering\n-------------------------------------------------------------------------------\n"Convictions are more dangerous enemies of truth than lies."  - F. W.\nNietzsche\n', 'From: jgreen
```

generate word counts for words in docs

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

[9]: (10, 881)

[10]: cv.get_feature_names()

[10]: ['083057',
       '10',
       '10mb',
       '11',
       '12',
       '120',
       '12mb',
       '14',
       '15',
       '16',
       '160',
       '161',
       '16899',
       '173',
```

get tfidf vector for first document

```
[13]: tfidf_transformer = TfidfTransformer()

      tf_idf_vector = tfidf_transformer.fit_transform(word_count_vector)

      feature_names = cv.get_feature_names()

      # get tfidf vector for first document
      first_document_vector = tf_idf_vector[2]

      # print the vector
      df = pd.DataFrame(first_document_vector.T.todense(), index = feature_names, columns = ["tfidf"])
      df.sort_values(by = ["tfidf"], ascending = False)

      df = pd.read_csv("abcnews-date-text.csv")

      data_text = df[["headline_text"]].astype("str")
      data_text.shape

[13]: (1082168, 1)
```

[15]:
```
import nltk
nltk.download("stopwords")
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\dhiva\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
```

[15]:
```
True
```

[16]:
```
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(x))
```

[17]:
```
data_text.head()
#stopw
```

[17]:

	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

[18]:
```
def word_count(x):
    terms = x.split()
    return len(terms)
data_text["word_count"] = data_text["Refined_headlines"].apply(lambda x: word_count(x))
```

[19]:
```
data_text.head()
```

[19]:

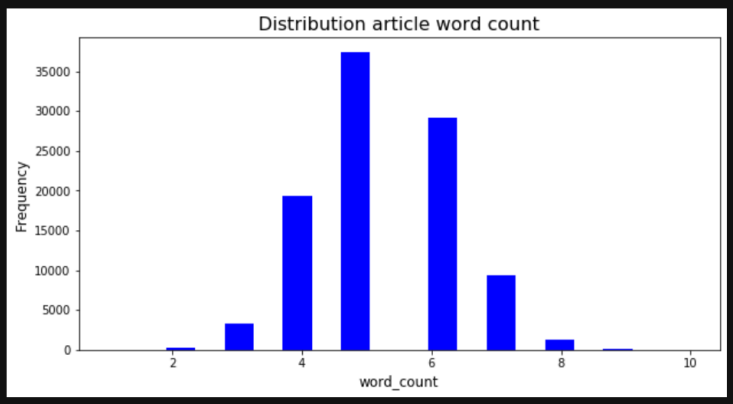
	headline_text	Refined_headlines	word_count
1	act fire witnesses must be aware of defamation	act fire witnesses must aware defamation	6
2	a g calls for infrastructure protection summit	g calls infrastructure protection summit	5
3	air nz staff in aust strike for pay rise	air nz staff aust strike pay rise	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

[20]:
```
data_text["word_count"].describe()
```

[20]:
```
count    100000.000000
mean          5.251110
std           1.035618
min           1.000000
25%           5.000000
50%           5.000000
75%           6.000000
max          10.000000
Name: word_count, dtype: float64
```

[21]:
```
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)
```

[21]:
```
Text(0.5, 0, 'word_count')
```



[22]:
```
headline_sentences = [''.join(text) for text in data_text["Refined_headlines"]]

vectorizer = CountVectorizer(max_features = 5000)
x_counts = vectorizer.fit_transform(headline_sentences)

transformer = TfidfTransformer()
x_tfidf = transformer.fit_transform(x_counts)
x_tfidf_norm = normalize(x_tfidf, norm = 'l1', axis = 1)
```

[23]:
```
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)
```

[24]:
```
get_nmf_topics(model, 30)
```

[24]:

	Topic #01	Topic #02	Topic #03	Topic #04	Topic #05
0	police	us	man	council	new
1	probe	iraq	charged	govt	zealand
2	investigate	killed	court	plan	laws
3	missing	troops	dies	water	president
4	search	iraqi	murder	urged	hospital
5	death	war	face	nsw	chief
6	shooting	says	car	says	set
7	fatal	baghdad	crash	call	record
8	car	soldiers	killed	boost	york
9	crash	soldier	hospital	wa	home
10	hunt	attack	jailed	qld	deal
11	seek	two	charges	funds	lead