Assignment 2

Submitted By: Aakash Patel (B00807065) Rishab Dhawan (B00826918)

1. Corpus Extraction:

In [1]:

```
#Import newsgroup corpus
from sklearn.datasets import fetch_20newsgroups
newsgroups_traindata = fetch_20newsgroups(subset='train')
```

In [2]:

```
from pprint import pprint
pprint(list(newsgroups_traindata.target_names))
```

```
['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']
```

In [3]:

```
newsgroups_traindata.filenames.size
```

Out[3]:

```
11314
```

In [4]:

```
newsgroups_traindata.target.size
```

Out[4]:

```
11314
```

In [5]:

```
corpus = ['alt.atheism','talk.religion.misc','comp.graphics','sci.space']
traindata = fetch_20newsgroups(subset='train', categories=corpus)
train_data = traindata.data
##print(train_data) ## this print statement has been commented due to excessively big output.
```

In [6]:

```
print(train_data[0])
```

```
From: rych@festival.ed.ac.uk (R Hawkes)
Subject: 3DS: Where did all the texture rules go?
Lines: 21

Hi,

I've noticed that if you only save a model (with all your mapping planes
positioned carefully) to a .3DS file that when you reload it after restarting
3DS, they are given a default position and orientation.  But if you save
to a .PRJ file their positions/orientation are preserved.  Does anyone
know why this information is not stored in the .3DS file?  Nothing is
explicitly said in the manual about saving texture rules in the .PRJ file.
I'd like to be able to read the texture rule information, does anyone have
the format for the .PRJ file?

Is the .CEL file format available from somewhere?

Rych

======================================================================
Rycharde Hawkes    email: rych@festival.ed.ac.uk
Virtual Environment Laboratory
Dept. of Psychology   Tel  : +44 31 650 3426
Univ. of Edinburgh   Fax  : +44 31 667 0150
======================================================================
```

1.a

Tokenization and POS tagging

Sentence tokenization

In [7]:

```
from nltk.tokenize import sent_tokenize
#Divide the text by sentences.
sent_list=[]
for text in train_data:
    sent=sent_tokenize(text)
    sent_list.append(sent)
print(sent_list[0])
```

```
['From: rych@festival.ed.ac.uk (R Hawkes)\nSubject: 3DS: Where did all the texture rules go?', "Lines: 21\n\nHi,\n\nI've noticed that if you only save a model (with all your mapping planes\npositioned carefully) to a .3DS file that when you reload it after restarting\n3DS, they are given a default position and orientation.", 'But if you save\nto a .PRJ file their positions/orientation are preserved.', 'Does anyone\nknow why this information is not stored in the .3DS file?', 'Nothing is\nexplicitly said in the manual about saving texture rules in the .PRJ file.', "I'd like to be able to read the texture rule information, does anyone have \nthe format for the .PRJ file?", 'Is the .CEL file format available from somewhere?', 'Rych\n\n======================================================================\nRycharde Hawkes\t\t\t\temail: rych@festival.ed.ac.uk\nVirtual Environment Laboratory\nDept.', 'of Psychology\t\t\tTel  : +44 31 650 3426\nUniv.', 'of Edinburgh\t\t\tFax  : +44 31 667 0150\n======================================================================']
```

Word Tokenization

In [8]:

```
from nltk.tokenize import word_tokenize
#more sophisticated than split by space to get the words.
tokens_list = []
for sub_list in sent_list:
    for sent in sub_list:
        tokens=word_tokenize(sent)
        tokens_list.append(tokens)
print(tokens_list[0])
```

```
print(tokens_list[1])
```

```
['From', ':', 'rych', '@', 'festival.ed.ac.uk', '(', 'R', 'Hawkes', ')', 'Subject', ':', '3DS', ':
', 'Where', 'did', 'all', 'the', 'texture', 'rules', 'go', '?']
['Lines', ':', '21', 'Hi', ',', 'I', "'ve", 'noticed', 'that', 'if', 'you', 'only', 'save', 'a', '
model', '(', 'with', 'all', 'your', 'mapping', 'planes', 'positioned', 'carefully', ')', 'to',
'a', '.3DS', 'file', 'that', 'when', 'you', 'reload', 'it', 'after', 'restarting', '3DS', ',', 'th
ey', 'are', 'given', 'a', 'default', 'position', 'and', 'orientation', '.']
```

POS Tagging

In [9]:

```
tag_list=[]
import nltk
for t in tokens_list:
    tagged=nltk.pos_tag(t)
    tag_list.append(tagged)
print(tag_list[0])
```

```
[('From', 'IN'), (':', ':'), ('rych', 'NN'), ('@', 'NN'), ('festival.ed.ac.uk', 'NN'), ('(', '('),
('R', 'NNP'), ('Hawkes', 'NNP'), (')', ')'), ('Subject', 'NN'), (':', ':'), ('3DS', 'CD'), (':',
':'), ('Where', 'WRB'), ('did', 'VBD'), ('all', 'PDT'), ('the', 'DT'), ('texture', 'NN'),
('rules', 'NNS'), ('go', 'VB'), ('?', '.')]
```

Preprocessing

Note: Following section involves basic cleaning (for better results). Other sections where its required we have performed cleaning in sequence.

In [10]:

```
import numpy as np
import pandas as pd
import re
import nltk
from nltk.corpus import stopwords
import string
```

Stop Words Filteration

In [11]:

```
stop_words=set(stopwords.words("english"))
print(stop_words)
```

```
{'who', 'while', 'at', "you'd", 'its', 'where', 'own', 'before', 'do', 'with', 'over', 'any', 'why
', 'whom', 'ourselves', 'itself', 'more', 'of', 'that', 'herself', 'being', "don't", 'once', 'a',
"wasn't", "weren't", 'to', 'off', 'into', 'those', 'hadn', 'theirs', 'further', 'not', 'down', 'sh
ould', 'hers', 'them', 'doing', 'same', 'shan', 'here', 'needn', 'there', "that'll", 'he', 'will',
"you've", "needn't", 'from', 'now', 'couldn', "shan't", 'haven', 'yourself', 'which', "hadn't", 'd
id', "couldn't", 'yours', 'i', 'just', 'weren', 'other', 'having', "didn't", 'was', 'until', "shou
ld've", 'aren', "you'll", 'ma', "she's", 'above', 'how', 'my', 'few', 'hasn', 'been', 'against', '
m', 'were', 'is', 'only', 'during', 'can', "haven't", 'on', 'about', 'again', 'you', 'out', 'no',
'as', 'some', 'both', 'wouldn', 'or', 'the', 'isn', "you're", "isn't", "hasn't", 'our', 'don', 'di
dn', 'doesn', 'has', 't', 'very', 'they', 'wasn', 'for', 'below', 'what', 'am', 'between', 'had',
'each', 'she', 're', 'yourselves', 'most', 'nor', "aren't", "doesn't", "shouldn't", 'him', 'then',
'be', 'themselves', 'and', 'under', 'ain', 'mustn', 'if', 'their', 'o', 'won', 've', 'we', 'an', '
so', 'because', 's', "won't", 'after', 'such', 'his', 'it', 'this', 'mightn', 'by', 'through', 'bu
t', 'these', 'myself', 'y', 'shouldn', 'have', "mightn't", 'when', 'your', 'himself', 'all',
"mustn't", 'does', 'me', 'are', 'too', 'than', "it's", 'in', 'up', 'd', "wouldn't", 'her', 'ours',
'll'}
```

In [12]:

```
filtered_token_list = []
for sub_list in tokens_list:
```

```
    for t in sub_list:
        if t not in stop_words:
            filtered_token_list.append(t)

##print(filtered_token_list) ## this print statement has been commented due to excessively big out
put.
```

Cleaning

In [13]:

```
import re
from string import punctuation

def _removeNonAscii(s): return "".join(i for i in s if ord(i)<128)

def _removeTags(s): return re.sub('<[^<]+?>','', s)

def _removeNumbers(text): return "".join(c for c in text if not c.isdigit())

def _removeNonAlpha(text): return "".join(c for c in text if c.isalpha())

def _removeHyperlinks(s): return re.sub(r"https\S+", " ", s)

def _removeNonAscii_(s): return  re.sub(r'[^\x00-\x7f]+', ' ', s)

def _removeNonLetter(s): return re.sub("[^., 'a-zA-Z]+", "", s)
```

In [14]:

```
cleaned_token_list = []
for token in filtered_token_list:
    s= _removeNumbers(token)
    if s is not "":
        s=_removeNonAlpha(s)
        if s is not "":
            (cleaned_token_list.append(s))
##print(cleaned_token_list) ## this print statement has been commented due to excessively big outp
ut.
```

Stemming

In [15]:

```
from nltk.stem.wordnet import WordNetLemmatizer
lem = WordNetLemmatizer()

from nltk.stem.porter import PorterStemmer
stem = PorterStemmer()

stem_list=[]
for word in cleaned_token_list:
    stem_list.append(stem.stem(word))

##print(stem_list)  ## this print statement has been commented due to excessively big output.
```

1.b

Bigram Collocation Extraction with four types of techniques:

In [16]:

```
bigrams = nltk.collocations.BigramAssocMeasures()
```

In [17]:

```
bigramFinder = nltk.collocations.BigramCollocationFinder.from_words(stem_list)
```

In [18]:

```
bigram_freq = bigramFinder.ngram_fd.items()
```

In [19]:

```
##bigram_freq  ## this print statement has been commented due to excessively big output.
```

1. Frequency with filter

In [20]:

```
bigramFreqTable = pd.DataFrame(list(bigram_freq), columns=['bigram','freq']).sort_values(by='freq',
ascending=False)
```

In [21]:

```
bigramFreqTable.head().reset_index(drop=True)
```

Out[21]:

	bigram	freq
0	(subject, Re)	1460
1	(In, articl)	1139
2	(I, m)	658
3	(line, In)	622
4	(I, nt)	620

In [22]:

```
bigramFreqTable[:20]
```

Out[22]:

	bigram	freq
80	(subject, Re)	1460
101	(In, articl)	1139
230	(I, m)	658
100	(line, In)	622
1095	(I, nt)	620
487	(I, think)	445
102	(articl, apr)	378
437	(line, nntppostinghost)	373
210	(write, In)	329
313	(I, would)	320
13	(I, ve)	305
1223	(ca, nt)	295
1266	(organ, univers)	291
946	(distribut, world)	230
2170	(It, s)	222
7121	(I, know)	218
392	(line, distribut)	186
1096	(nt, know)	168
2797	(write, I)	166

	bigram	freq
50	(l, d)	156

In [23]:

```
#get english stopwords
en_stopwords = set(stopwords.words('english'))
```

In [24]:

```
#function to filter for ADJ/NN bigrams
def rightTypes(ngram):
    if '-pron-' in ngram or '' in ngram or ' 'in ngram or 't' in ngram:
        return False
    for word in ngram:
        if word in en_stopwords:
            return False
    acceptable_types = ('JJ', 'JJR', 'JJS', 'NN', 'NNS', 'NNP', 'NNPS')
    second_type = ('NN', 'NNS', 'NNP', 'NNPS')
    tags = nltk.pos_tag(ngram)
    if tags[0][1] in acceptable_types and tags[1][1] in second_type:
        return True
    else:
        return False
```

In [25]:

```
#filter bigrams
filtered_bi = bigramFreqTable[bigramFreqTable.bigram.map(lambda x: rightTypes(x))]
```

In [26]:

```
filtered_bi[:20]
```

Out[26]:

	bigram	freq
80	(subject, Re)	1460
102	(articl, apr)	378
437	(line, nntppostinghost)	373
1266	(organ, univers)	291
946	(distribut, world)	230
392	(line, distribut)	186
947	(world, nntppostinghost)	128
3817	(sandvik, newtonapplecom)	126
3819	(kent, sandvik)	124
6120	(xnewsread, tin)	111
391	(usa, line)	110
431	(comput, scienc)	104
16665	(space, station)	102
788	(henri, spencer)	96
4731	(jon, livesey)	94
786	(henri, zootorontoedu)	92
37	(doe, anyon)	92
4729	(livesey, solntzewpdsgicom)	92
4745	(keith, ccocaltechedu)	90
6121	(tin, version)	90

In [27]:

```
freq_bi = filtered_bi[:20].bigram.values
```

2. PMI

In [28]:

```
bigramFinder.apply_freq_filter(20)
```

In [29]:

```
bigramPMITable = pd.DataFrame(list(bigramFinder.score_ngrams(bigrams.pmi)),
columns=['bigram','PMI']).sort_values(by='PMI', ascending=False)
```

In [30]:

```
bigramPMITable[:20]
```

Out[30]:

	bigram	PMI
0	(steinn, sigurdsson)	14.129701
1	(kjenk, gothamcityjscnasagov)	13.998456
2	(coegalon, larcnasagov)	13.937056
3	(sank, manhattan)	13.931342
4	(comm, aucun)	13.878162
5	(carnegi, mellon)	13.714663
6	(fait, comm)	13.676528
7	(pharvey, quackkfucom)	13.664037
8	(blew, bronx)	13.664037
9	(bake, timmon)	13.589132
10	(mlee, postroyalroadsca)	13.567822
11	(originalsend, isu)	13.522018
12	(isu, vacationvenaricscmuedu)	13.467570
13	(chapel, hill)	13.416224
14	(sunris, sunset)	13.358520
15	(cookamunga, tourist)	13.274091
16	(researchcanonozau, enzo)	13.271057
17	(alaska, fairbank)	13.164466
18	(MS, telo)	13.129701
19	(sysmgr, kingengumdedu)	13.095753

In [31]:

```
pmi_bi = bigramPMITable[:20].bigram.values
```

3. T-test with filter

In [32]:

```
bigramTtable = pd.DataFrame(list(bigramFinder.score_ngrams(bigrams.student_t)), columns=['bigram',
't']).sort_values(by='t', ascending=False)
```

In [33]:

```
bigramTtable.head()
```

Out[33]:

	bigram	t
0	(subject, Re)	37.954492
1	(In, articl)	33.541061
2	(I, m)	25.012786
3	(line, In)	24.467547
4	(I, nt)	22.467302

In [34]:

```
filteredT_bi = bigramTtable[bigramTtable.bigram.map(lambda x: rightTypes(x))]
```

In [35]:

```
filteredT_bi[:20]
```

Out[35]:

	bigram	t
0	(subject, Re)	37.954492
6	(articl, apr)	19.311449
7	(line, nntppostinghost)	19.013942
11	(organ, univers)	16.679715
13	(distribut, world)	15.115983
15	(line, distribut)	13.422339
21	(sandvik, newtonapplecom)	11.217845
22	(world, nntppostinghost)	11.182723
23	(kent, sandvik)	11.127318
26	(xnewsread, tin)	10.531435
27	(usa, line)	10.362541
29	(comput, scienc)	10.131995
30	(space, station)	10.036208
34	(henri, spencer)	9.790990
36	(jon, livesey)	9.686311
37	(livesey, solntzewpdsgicom)	9.584644
38	(henri, zootorontoedu)	9.583951
39	(doe, anyon)	9.564956
41	(keith, ccocaltechedu)	9.475762
42	(version, PL)	9.474144

In [36]:

```
t_bi = filteredT_bi[:20].bigram.values
```

4. Chi-Sq-test

In [37]:

```
bigramChiTable = pd.DataFrame(list(bigramFinder.score_ngrams(bigrams.chi_sq)),
columns=['bigram','chi-sq']).sort_values(by='chi-sq', ascending=False)
```

In [38]:

```
bigramChiTable.head(20)
```

Out[38]:

	bigram	chi-sq
0	(alink, ksand)	376429.000000
1	(steinn, sigurdsson)	376429.000000
2	(daric, yoyoccmonasheduau)	368418.893497
3	(cookamunga, tourist)	366521.999904
4	(vaxvm, vnew)	363666.762419
5	(carnegi, mellon)	362984.142788
6	(coegalon, larcnasagov)	360743.499941
7	(kmr, pocwruedu)	356348.999286
8	(mlee, postroyalroadsca)	352141.386953
9	(kjenk, gothamcityjscnasagov)	343694.217289
10	(sank, manhattan)	328070.715255
11	(allan, schneider)	314642.459003
12	(newssoftwar, vaxvm)	309379.343936
13	(bobb, viceicotekcom)	308853.331380
14	(originalsend, isu)	305843.687163
15	(comm, aucun)	301139.199787
16	(mango, csumdedu)	295229.803005
17	(isu, vacationvenaricscmuedu)	294514.999584
18	(prb, accessdigexcom)	293331.891940
19	(pharvey, quackkfucom)	285561.516931

In [39]:

```
chi_bi = bigramChiTable[:20].bigram.values
```

1.c

Comparison of the results and respective insights:

In [40]:

```
bigramsCompare = pd.DataFrame([freq_bi, pmi_bi, t_bi, chi_bi]).T
```

In [41]:

```
bigramsCompare.columns = ['Frequency With Filter', 'PMI', 'T-test With Filter', 'Chi-Sq Test']
```

In [42]:

```
bigramsCompare
```

Out[42]:

	Frequency With Filter	PMI	T-test With Filter	Chi-Sq Test
0	(subject, Re)	(steinn, sigurdsson)	(subject, Re)	(alink, ksand)
1	(articl, apr)	(kjenk, gothamcityjscnasagov)	(articl, apr)	(steinn, sigurdsson)
2	(line, nntppostinghost)	(coegalon, larcnasagov)	(line, nntppostinghost)	(daric, yoyoccmonasheduau)
3	(organ, univers)	(sank, manhattan)	(organ, univers)	(cookamunga, tourist)
4	(distribut, world)	(comm, aucun)	(distribut, world)	(vaxvm, vnew)
5	(line, distribut)	(carnegi, mellon)	(line, distribut)	(carnegi, mellon)
6	(world, nntppostinghost)	(fait, comm)	(sandvik, newtonapplecom)	(coegalon, larcnasagov)

	Frequency With Filter	PMI	T-test With Filter	Chi-Sq Test
6	(world, nntppostinghost)	(fait, comm)	(sandvik, newtonapplecom)	(coegaion, iarchasagov)
7	(sandvik, newtonapplecom)	(pharvey, quackkfucom)	(world, nntppostinghost)	(kmr, pocwruedu)
8	(kent, sandvik)	(blew, bronx)	(kent, sandvik)	(mlee, postroyalroadsca)
9	(xnewsread, tin)	(bake, timmon)	(xnewsread, tin)	(kjenk, gothamcityjscnasagov)
10	(usa, line)	(mlee, postroyalroadsca)	(usa, line)	(sank, manhattan)
11	(comput, scienc)	(originalsend, isu)	(comput, scienc)	(allan, schneider)
12	(space, station)	(isu, vacationvenaricscmuedu)	(space, station)	(newssoftwar, vaxvm)
13	(henri, spencer)	(chapel, hill)	(henri, spencer)	(bobb, viceicotekcom)
14	(jon, livesey)	(sunris, sunset)	(jon, livesey)	(originalsend, isu)
15	(henri, zootorontoedu)	(cookamunga, tourist)	(livesey, solntzewpdsgicom)	(comm, aucun)
16	(doe, anyon)	(researchcanonozau, enzo)	(henri, zootorontoedu)	(mango, csumdedu)
17	(livesey, solntzewpdsgicom)	(alaska, fairbank)	(doe, anyon)	(isu, vacationvenaricscmuedu)
18	(keith, ccocaltechedu)	(MS, telo)	(keith, ccocaltechedu)	(prb, accessdigexcom)
19	(tin, version)	(sysmgr, kingengumdedu)	(version, PL)	(pharvey, quackkfucom)

Overlap In Techniques-

Here we see that PMI and Chi-Sq test give pretty accurate results even without filters.Considering the rank comparison among the techniques we find that Frequency with filter and T-test with filter shows similarity in almost 75% overlap of the results where as PMI and Chi-Square have less than 30% matches. Thus taking union among the results of T-test and frequency with filter makes sense to a certain extent. However, PMI and Chi-Square test are must be a distinguished test attributes to consider during the bigram collocation extraction from the corpus.

2. SVM (Support Vector Machines) and NB (Naive Bayes) for Text Classification:

2.a Cleaning for Bag of words:

In [43]:

```
# Cleans token to remove numbers and non-alphabets
def cleantoken(token_list):
    cleaned_token_list=[]
    for token in token_list:
        s= _removeNumbers(token)
        if s is not "":
            s=_removeNonAlpha(s)
            if s is not "":
                (cleaned_token_list.append(s))
    return cleaned_token_list

# Returns stemmed list from given list of tokens
def stemmer(cleaned_token_list):
    stem_list=[]
    for word in cleaned_token_list:
        stem_list.append(stem.stem(word))
    return stem_list
```

In [44]:

```
article_token_list=[]
# Main list consisting of List of (bag of tokens of articles)
for article in sent_list:
    token_bag=[]
    token_sent_list=[]
    bag_sent=""
    for sent in article:
        #tokenize
        tokens=word_tokenize(sent)
        #removing stop words
        for t in tokens:
            if t not in stop_words:
                token_bag.append(t)
        #cleaning
        token_bag=cleantoken(token_bag)
```

```
        #stemming
        token_bag = stemmer(token_bag)
        #forming sentence with space
        sent=""
        for t in token_bag:
            sent=sent+t+" "
        bag_sent+=sent


    article_token_list.append(bag_sent)
```

In [45]:

```
article_token_list[0]
```

Out[45]:

```
'from rych festivaledacuk R hawk subject DS where textur rule go from rych festivaledacuk R hawk s
ubject DS where textur rule go line Hi I ve notic save model map plane posit care DS file reload
restart DS given default posit orient from rych festivaledacuk R hawk subject DS where textur rule
go line Hi I ve notic save model map plane posit care DS file reload restart DS given default
posit orient but save prj file positionsorient preserv from rych festivaledacuk R hawk subject DS
where textur rule go line Hi I ve notic save model map plane posit care DS file reload restart DS
given default posit orient but save prj file positionsori preserv doe anyon know inform store DS f
ile from rych festivaledacuk R hawk subject DS where textur rule go line Hi I ve notic save model
map plane posit care DS file reload restart DS given default posit orient but save prj file
positionsori preserv doe anyon know inform store DS file noth explicitli said manual save textur r
ule prj file from rych festivaledacuk R hawk subject DS where textur rule go line Hi I ve notic sa
ve model map plane posit care DS file reload restart DS given default posit orient but save prj fi
le positionsori preserv doe anyon know inform store DS file noth explicitli said manual save
textur rule prj file I d like abl read textur rule inform anyon format prj file from rych festival
edacuk R hawk subject DS where textur rule go line Hi I ve notic save model map plane posit care
DS file reload restart DS given default posit orient but save prj file positionsori preserv doe an
yon know inform store DS file noth explicitli said manual save textur rule prj file I d like abl r
ead textur rule inform anyon format prj file Is cel file format avail somewher from rych
festivaledacuk R hawk subject DS where textur rule go line Hi I ve notic save model map plane
posit care DS file reload restart DS given default posit orient but save prj file positionsori pre
serv doe anyon know inform store DS file noth explicitli said manual save textur rule prj file I d
like abl read textur rule inform anyon format prj file Is cel file format avail somewh rych rychar
d hawk email rych festivaledacuk virtual environ laboratori dept from rych festivaledacuk R hawk s
ubject DS where textur rule go line Hi I ve notic save model map plane posit care DS file reload
restart DS given default posit orient but save prj file positionsori preserv doe anyon know inform
store DS file noth explicitli said manual save textur rule prj file I d like abl read textur rule
inform anyon format prj file Is cel file format avail somewh rych rychard hawk email rych
festivaledacuk virtual environ laboratori dept psycholog tel univ from rych festivaledacuk R hawk
subject DS where textur rule go line Hi I ve notic save model map plane posit care DS file reload
restart DS given default posit orient but save prj file positionsori preserv doe anyon know inform
store DS file noth explicitli said manual save textur rule prj file I d like abl read textur rule
inform anyon format prj file Is cel file format avail somewh rych rychard hawk email rych
festivaledacuk virtual environ laboratori dept psycholog tel univ edinburgh fax '
```

2 b. TF-IDF

Extracting features out of text files

In [46]:

```
from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(article_token_list)
X_train_counts.shape
```

Out[46]:

```
(2034, 25290)
```

TF-IDF Weighted Vector Representation

In [47]:

```
from sklearn.feature_extraction.text import TfidfTransformer
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
X_train_tfidf.shape
```

Out[47]:

```
(2034, 25290)
```

2.c Model analysis of training and test data set

In [48]:

```
traindata.target
```

Out[48]:

```
array([1, 3, 2, ..., 1, 0, 1], dtype=int64)
```

In [49]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X_train_tfidf, traindata.target, test_size=0.3, random_state=42)
```

In [50]:

```
from sklearn.linear_model import SGDClassifier
from sklearn import svm
from sklearn import metrics

#Create a svm Classifier
clf = svm.SVC(kernel='linear')
clf.fit(X_train, y_train)
y_test_pred = clf.predict(X_test)
y_train_pred = clf.predict(X_train)
print("SVM :")
print("Test Accuracy:",metrics.accuracy_score(y_test, y_test_pred))
print("Train Accuracy:",metrics.accuracy_score(y_train, y_train_pred))
```

```
SVM :
Test Accuracy: 0.9623567921440261
Train Accuracy: 0.9978917779339423
```

In [51]:

```
from sklearn.metrics import confusion_matrix
def confusionMatrix(clfname, y_train, y_train_pred, y_test, y_test_pred):
    print("Confusion Matrix for "+clfname+":-")
    cm_train = confusion_matrix(y_train, y_train_pred)
    cm_test = confusion_matrix(y_test, y_test_pred)
    print("Training set confusion matrix : \n"+str(cm_train))
    print("Test set confusion matrix : \n"+str(cm_test))
```

In [52]:

```
confusionMatrix("SVM", y_train, y_train_pred, y_test, y_test_pred)
```

```
Confusion Matrix for SVM:-
Training set confusion matrix :
[[331   0   0   2]
 [  0 410   0   0]
 [  0   0 412   0]
 [  1   0   0 267]]
Test set confusion matrix :
[[136   1   2   8]
 [  0 174   0   0]
 [  0   5 176   0]
 [  2   4   1 102]]
```

In [53]:

```
from sklearn.naive_bayes import MultinomialNB
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Model Generation Using Multinomial Naive Bayes
clf = MultinomialNB().fit(X_train, y_train)
clf.fit(X_train, y_train)
y_test_pred = clf.predict(X_test)
y_train_pred = clf.predict(X_train)
print("MultinomialNB :")
print("Test Accuracy:",metrics.accuracy_score(y_test, y_test_pred))
print("Train Accuracy:",metrics.accuracy_score(y_train, y_train_pred))
```

```
MultinomialNB :
Test Accuracy: 0.911620294599018
Train Accuracy: 0.9648629655657063
```

In [54]:

```
confusionMatrix("MultinomialNB", y_train, y_train_pred, y_test, y_test_pred)
```

```
Confusion Matrix for MultinomialNB:-
Training set confusion matrix :
[[333   0   0   0]
 [  0 410   0   0]
 [  0   0 412   0]
 [ 41   1   8 218]]
Test set confusion matrix :
[[145   0   1   1]
 [  0 169   5   0]
 [  0   2 179   0]
 [ 35   1   9  64]]
```

Clearly here we visulaise that SVM is having better accuracy compared to the Naive Bayes. Its because NB treats the corpus feature set as independendent entities seprarating them by geometrical mathematics distributions; whereas Support vector mahines also consider certain interactions between them toa certain degree. moreover, for smaller data sets NB out performs SVM butsince our data set is quite big news corpus with humungous size, SVM provides better results compared to the NB. From theoretical point of view, both approaches are different in cosiderations. One is probibalistic(NB) which is better if we consider on chunks of content data and the other is geometrical(SVM) which is better at full sized content. therefore For the size of data we have NB will be better if we consider only chunks of data, but SVM will beat it when the we consider the complete data set size.

In [55]:

```
clf = svm.SVC(kernel='poly')
clf.fit(X_train, y_train)
y_test_pred = clf.predict(X_test)
y_train_pred = clf.predict(X_train)
print("SVM with Poly kernel :")
print("Test Accuracy:",metrics.accuracy_score(y_test, y_test_pred))
print("Train Accuracy:",metrics.accuracy_score(y_train, y_train_pred))
```

```
c:\users\aakash patel\appdata\local\programs\python\python37\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The default value of gamma will change from 'auto' to 'scale' in version 0.22 to account better for unscaled features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

```
SVM with Poly kernel :
Test Accuracy: 0.29623567921440264
Train Accuracy: 0.28952916373858045
```

In [56]:

```
clf = svm.SVC(kernel='rbf')
clf.fit(X_train, y_train)
y_test_pred = clf.predict(X_test)
y_train_pred = clf.predict(X_train)
```

```
print("SVM with rbf kernel :")
print("Test Accuracy:",metrics.accuracy_score(y_test, y_test_pred))
print("Train Accuracy:",metrics.accuracy_score(y_train, y_train_pred))
```

```
c:\users\aakash patel\appdata\local\programs\python\python37\lib\site-
packages\sklearn\svm\base.py:193: FutureWarning: The default value of gamma will change from
'auto' to 'scale' in version 0.22 to account better for unscaled features. Set gamma explicitly to
'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

```
SVM with rbf kernel :
Test Accuracy: 0.29623567921440264
Train Accuracy: 0.28952916373858045
```

Changing Kernel decreases the accuracy. Linear kernel has maximum risk of underfitting and lowest risk of overfitting. Ability to fit any data is also lowest for Linear compared to other both. Due to all these linear kernel performs better.

2.d

Noun Tagging & Extraction function

In [57]:

```
def nounTagger(tokens_list):
    tag_noun_list=[]
    nouns=[]
    tagged=nltk.pos_tag(tokens_list)
    for s in tagged:
        if(s[1] in ['NN', 'NNS', 'NNP','NNPS']):
            nouns.append(s[0])     #only taking the token (without tag)
    return nouns
```

POS tagging prior to cleaning

In [58]:

```
#first article
print(sent_list[0])
```

```
['From: rych@festival.ed.ac.uk (R Hawkes)\nSubject: 3DS: Where did all the texture rules go?', "Li
nes: 21\n\nHi,\n\nI've noticed that if you only save a model (with all your mapping
planes\npositioned carefully) to a .3DS file that when you reload it after restarting\n3DS, they a
re given a default position and orientation.", 'But if you save\nto a .PRJ file their
positions/orientation are preserved.', 'Does anyone\nknow why this information is not stored in th
e .3DS file?', 'Nothing is\nexplicitly said in the manual about saving texture rules in the .PRJ f
ile.', "I'd like to be able to read the texture rule information, does anyone have \nthe format fo
r the .PRJ file?", 'Is the .CEL file format available from somewhere?',
'Rych\n\n=================================================================\nRycharde
Hawkes\t\t\t\temail: rych@festival.ed.ac.uk\nVirtual Environment Laboratory\nDept.', 'of
Psychology\t\t\tTel  : +44 31 650 3426\nUniv.', 'of Edinburgh\t\t\tFax  : +44 31 667
0150\n=======================================================================']
```

In [59]:

```
article_token_list=[]
# Maintain list consisting of List of (bag of tokens of articles)
for article in sent_list:
    token_bag=[]
    token_sent_list=[]
    bag_sent=""
    for sent in article:
        #tokenize
        tsent=[]
        tokens=word_tokenize(sent)
        #tagging nouns
        tokens=nounTagger(tokens)
        #removing stop words
```

```
        for t in tokens:
            if t not in stop_words:
                tsent.append(t)

        #cleaning
        tsent=cleantoken(tsent)
        #stemming
        tsent=stemmer(tsent)
        #forming sentence with space
        sent=""
        for t in tsent:
            sent=sent+t+" "
        bag_sent+=sent

    article_token_list.append(bag_sent)
```

In [60]:

```
#processed first article (merged)
print(article_token_list[0])
```

```
rych festivaledacuk R hawk subject textur rule line Hi model map plane DS file default posit
orient prj file positionsorient doe anyon inform DS file noth textur rule prj file textur rule inf
orm anyon format prj file file format rych rychard hawk email rych virtual environ laboratori dept
psycholog tel univ edinburgh fax
```

Feature Extraction

In [61]:

```
from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(article_token_list)
X_train_counts.shape
```

Out[61]:

```
(2034, 19686)
```

TF-IDF Vector Representation

In [62]:

```
from sklearn.feature_extraction.text import TfidfTransformer
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
X_train_tfidf.shape
```

Out[62]:

```
(2034, 19686)
```

In [63]:

```
traindata.target
```

Out[63]:

```
array([1, 3, 2, ..., 1, 0, 1], dtype=int64)
```

In [64]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X_train_tfidf, traindata.target, test_size=0.3, random_state=42)
```

Support Vector Machine

In [65]:

```
from sklearn.linear_model import SGDClassifier
from sklearn import svm
from sklearn import metrics

#Create a svm Classifier
clf = svm.SVC(kernel='linear')
clf.fit(X_train, y_train)
y_test_pred = clf.predict(X_test)
y_train_pred = clf.predict(X_train)
print("SVM :")
print("Test Accuracy:",metrics.accuracy_score(y_test, y_test_pred))
print("Train Accuracy:",metrics.accuracy_score(y_train, y_train_pred))
```

```
SVM :
Test Accuracy: 0.9509001636661211
Train Accuracy: 0.9985945186226283
```

Confusion Matrix

In [66]:

```
from sklearn.metrics import confusion_matrix
def confusionMatrix(clfname, y_train, y_train_pred, y_test, y_test_pred):
    print("Confusion Matrix for "+clfname+":-")
    cm_train = confusion_matrix(y_train, y_train_pred)
    cm_test = confusion_matrix(y_test, y_test_pred)
    print("Training set confusion matrix : \n"+str(cm_train))
    print("Test set confusion matrix : \n"+str(cm_test))
```

In [67]:

```
confusionMatrix("SVM", y_train, y_train_pred, y_test, y_test_pred)
```

```
Confusion Matrix for SVM:-
Training set confusion matrix :
[[332   0   0   1]
 [  0 410   0   0]
 [  0   0 412   0]
 [  1   0   0 267]]
Test set confusion matrix :
[[133   2   1  11]
 [  0 174   0   0]
 [  0   6 174   1]
 [  5   3   1 100]]
```

Multinomial Naive Bayes

In [68]:

```
from sklearn.naive_bayes import MultinomialNB
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Model Generation Using Multinomial Naive Bayes
clf = MultinomialNB().fit(X_train, y_train)
clf.fit(X_train, y_train)
y_test_pred = clf.predict(X_test)
y_train_pred = clf.predict(X_train)
print("MultinomialNB :")
print("Test Accuracy:",metrics.accuracy_score(y_test, y_test_pred))
print("Train Accuracy:",metrics.accuracy_score(y_train, y_train_pred))
```

```
MultinomialNB :
Test Accuracy: 0.9247135842880524
Train Accuracy: 0.977512297962052
```

Confusion Matrix

In [69]:

```
confusionMatrix("MultinomialNB", y_train, y_train_pred, y_test, y_test_pred)
```

```
Confusion Matrix for MultinomialNB:-
Training set confusion matrix :
[[332   0   0   1]
 [  0 410   0   0]
 [  0   0 412   0]
 [ 27   1   3 237]]
Test set confusion matrix :
[[143   0   2   2]
 [  0 168   6   0]
 [  0   4 177   0]
 [ 25   1   6  77]]
```

Other Kernels

In [70]:

```
clf = svm.SVC(kernel='poly')
clf.fit(X_train, y_train)
y_test_pred = clf.predict(X_test)
y_train_pred = clf.predict(X_train)
print("SVM with Poly kernel :")
print("Test Accuracy:",metrics.accuracy_score(y_test, y_test_pred))
print("Train Accuracy:",metrics.accuracy_score(y_train, y_train_pred))
```

```
c:\users\aakash patel\appdata\local\programs\python\python37\lib\site-
packages\sklearn\svm\base.py:193: FutureWarning: The default value of gamma will change from
'auto' to 'scale' in version 0.22 to account better for unscaled features. Set gamma explicitly to
'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

```
SVM with Poly kernel :
Test Accuracy: 0.29623567921440264
Train Accuracy: 0.28952916373858045
```

In [71]:

```
clf = svm.SVC(kernel='rbf')
clf.fit(X_train, y_train)
y_test_pred = clf.predict(X_test)
y_train_pred = clf.predict(X_train)
print("SVM with rbf kernel :")
print("Test Accuracy:",metrics.accuracy_score(y_test, y_test_pred))
print("Train Accuracy:",metrics.accuracy_score(y_train, y_train_pred))
```

```
c:\users\aakash patel\appdata\local\programs\python\python37\lib\site-
packages\sklearn\svm\base.py:193: FutureWarning: The default value of gamma will change from
'auto' to 'scale' in version 0.22 to account better for unscaled features. Set gamma explicitly to
'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

```
SVM with rbf kernel :
Test Accuracy: 0.29623567921440264
Train Accuracy: 0.28952916373858045
```

Accuracy comparison:

On comparing the outputs in SVM when we compare output with nouns only the test accuracy dereases, whereas the training test accuracy increases. Following are the results: SVM (all type of words): Test Accuracy: 0.9623567921440261 Train Accuracy: 0.9978917779339423

SVM (only nouns): Test Accuracy: 0.9509001636661211 Train Accuracy: 0.9985945186226283

On comparing the outputs in Multinomial NB when we compare output with nouns only the test accuracy increases, and the training test accuracy also increases.

MultinomialNB : Test Accuracy: 0.911620294599018 Train Accuracy: 0.9648629655657063

MultinomialNB : Test Accuracy: 0.9247135842880524 Train Accuracy: 0.977512297962052

Vocabulary Size Comparison:

The original vocabulary size is depicted by the tf idf vector size which is as follows: (2034, 25290)

Next when we consider only nouns, the size will be given again by te tfidf vector size which is as follows: (2034, 19686)

Clearly considering nouns only will possess a smaller vocabulary compared to the original corpus size considering all dofferent types of vocabulary in the text.

References:

https://github.com/nicharuc/Collocations/blob/master/Collocations.i

https://stackoverflow.com/questions/12851791/removing-numbers-from-string/12856384

https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python

https://www.datacamp.com/community/tutorials/text-analytics-beginners-nltk]]

http://www.semspirit.com/artificial-intelligence/machine-learning/classification/support-vector-machine/support-vector-machine-classification-in-python/

https://stackoverflow.com/questions/33778297/support-vector-machine-kernel-types