TRIVENI ASHOK NAIK
SUID: 406461644

# *CIS - 668 Natural Language Processing*
# *Assignment 1*

## Task 1 - Choosing the data and processing

**1. (10 pts) Choose or Collect Appropriate Data: the two documents should be sufficiently different to yield good questions and of sufficient length for the word frequency and bigram lists to be useful.**

1. I have collected the data from Amazon and compared the reviews of the **Automotive products and the Musical instruments.**

**Additional merit**

2. I collected the data i.e the Amazon Product reviews from the Julian McAuley, UCSD. He has collected many product reviews of variety of items present. The link to this is http://jmcauley.ucsd.edu/data/amazon/

3. I downloaded the data from the above mentioned website. The data when downloaded was in the form of json. In order to process it I converted it to .txt. So that the computation would be easy.

4. To load the own data i.e the Amazon Reviews, I followed the following steps.

```
import nltk
from nltk.corpus import PlaintextCorpusReader
corpus_root = '/Users/triveninaik/Desktop'
wordlists = PlaintextCorpusReader(corpus_root, 'automotive.txt')
wordlists.fileids()
```

5. The Automotive reviews totally consists of 20,473 reviews and the musical instruments consists of 10,261 reviews.

6. These reviews consists of the reviewer name and the reviewer ID.

7. Automotive reviews consists of reviews of automotives like car, battery, cables , kinds of cable used, resistance of the vehicle and certain talks about it.

8. Musical Instruments reviews consists of the sound quality of the instruments, filters used, and the discussion about the different kind of musical instruments used like the guitar, piano.

TRIVENI ASHOK NAIK
SUID: 406461644

**2. (30 pts) Process each document and produce the frequency, bigram frequency and bigram PMI score lists, with processing steps chosen to produce lists suitable for analysis of your question.**

## Task 2 - Examining the text in the documents and processing of the text

For Document 1 – Automotive Products

1. I first started with the reviews of the automotive products.
2. First I tokenized the text using the nltk.word_tokenize.
3. Then I found out the length of the words and it was 3585688.
4. I found out the top 50 most frequent words. They are given below

```
``       329056
''       297769
,        261622
:        186282
the      93734
.        92379
i        51738
and      51541
to       50739
a        48269
it       45093
0        26769
is       26150
of       25552
this     24661
for      23149
overall          20921
helpful          20614
[        20598
]        20588
summary          20500
}        20491
{        20485
asin     20477
reviewerid       20473
reviewtext       20473
unixreviewtime           20473
reviewtime       20473
reviewername     20260
in       20189
on       19818
my       19286
you      17792
that     17493
with     16011
have     14148
5.0      13931
but      12934
```

```
not      12610
as       11028
n't      10577
!        10020
are      9973
was      9786
's       9433
2013     9212
they     9157
great    9021
so       8826
use      8481
```

5. I used the below line to get all the words in lower case.
   automotivewords = [w.lower() for w in automotivetokens]
   The output is as follows.

```
['{', '``', 'reviewerid', "'", ':', '``', 'a3f73sc1ly51oo', "'", '
,', '``', 'asin', "'", ':', '``', 'b00002243x', "'", ',', '``', 'r
eviewername', "'", ':', '``', 'alan', 'montgomery', "'", ',', '``'
, 'helpful', "'", ':', '[', '4', ',', '4', ']', ',', '``', 'reviewt
ext', "'", ':', '``', 'i', 'needed', 'a', 'set', 'of', 'jumper', 'c
ables', 'for', 'my', 'new', 'car', 'and', 'these', 'had', 'good', 'r
eviews', 'and', 'were', 'at', 'a', 'good', 'price', '.', 'they', 'ha
ve', 'been', 'used', 'a', 'few', 'times', 'already', 'and', 'do', 'w
hat', 'they', 'are', 'supposed', 'to', '-', 'no', 'complaints', 'the
re.what', 'i', 'will', 'say', 'is', 'that', '12', 'feet', 'really',
'is', "n't", 'an', 'ideal', 'length', '.', 'sure', ',', 'if', 'you',
'pull', 'up', 'front', 'bumper', 'to', 'front', 'bumper', 'they', 'a
re']
```

6. I applied a function to remove all the non-alphabetical words and also determine the len gth of such words.

## Additional merit

7. I created a new list of stopwords and added three extra stopwords to the existing steps. The following are the steps.

```
newstopwords = stopwords.append("can't, won't, n't, with")
print('number stopwords:', len(stopwords))
print(stopwords)
```

```
number stopwords: 157
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you',
'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'hims
elf', 'she', 'her', 'hers', 'herself', 'it', 'its', 'itself', 'they'
, 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', '
whom', 'this', 'that', 'these', 'those', 'am', 'is', 'are', 'was', '
were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do',
```

```
'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or',
'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with',
'about', 'against', 'between', 'into', 'through', 'during', 'before'
, 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out'
, 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once',
'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both'
, 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor
', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', '
t', 'can', 'will', 'just', 'don', 'should', 'now', 'd', 'll', 'm', '
o', 're', 've', 'y', 'ain', 'aren', 'couldn', 'didn', 'doesn', 'hadn
', 'hasn', 'haven', 'isn', 'ma', 'mightn', 'mustn', 'needn', 'shan',
'shouldn', 'wasn', 'weren', 'won', 'wouldn', ‘can't’, ‘won't ‘,‘n't’
,’with’]
```

8. The frequency distribution of top 50 words after removing the stopwords is given below

```
('overall', 20921)
('helpful', 20614)
('summary', 20500)
('asin', 20477)
('reviewerid', 20473)
('reviewtext', 20473)
('unixreviewtime', 20473)
('reviewtime', 20473)
('reviewername', 20260)
("'s", 9433)
('great', 9021)
('use', 8481)
('car', 8192)
('good', 7842)
('one', 7385)
('product', 6447)
('well', 6319)
('like', 6041)
('would', 5274)
('works', 5229)
('used', 5137)
('get', 4911)
('easy', 4272)
('time', 3864)
('work', 3805)
('much', 3546)
('battery', 3453)
('really', 3236)
('better', 3193)
('nice', 3099)
('also', 3069)
('using', 3029)
('price', 3009)
('little', 2970)
('need', 2962)
('oil', 2767)
('light', 2692)
('quality', 2641)
("'ve", 2632)
```

```
('water', 2596)
('fit', 2480)
('even', 2460)
('bought', 2407)
('made', 2305)
('new', 2247)
('best', 2243)
('long', 2238)
('filter', 2226)
('make', 2204)
```

8. The list of the top 50 bigram frequenices is given below

```
(("''", ':'), 0.05132766710321701)
((',', '``'), 0.04561774476753137)
((':', '``'), 0.03419845786917322)
(("''", ','), 0.022281637443079264)
(('``', ','), 0.006243711109276658)
(('``', 'overall'), 0.005716615611843529)
((':', '['), 0.005712711200751432)
(('helpful', "''"), 0.005711874541231697)
(('overall', "''"), 0.005711595654725119)
(('``', 'helpful'), 0.00571131676821854)
((']', ','), 0.005711037881711962)
(('``', 'summary'), 0.005711037881711962)
(("''", '}'), 0.005709643449179069)
(('``', 'asin'), 0.005709643449179069)
(('``', 'reviewerid'), 0.005709643449179069)
(('``', 'reviewtext'), 0.005709643449179069)

(('``', 'reviewtime'), 0.005709643449179069)
(('``', 'unixreviewtime'), 0.005709643449179069)
(('asin', "''"), 0.005709643449179069)
(('reviewerid', "''"), 0.005709643449179069)
(('reviewtext', "''"), 0.005709643449179069)
(('reviewtime', "''"), 0.005709643449179069)
(('summary', "''"), 0.005709643449179069)
(('unixreviewtime', "''"), 0.005709643449179069)
(('{', '``'), 0.005709643449179069)
(('}', '{'), 0.005709364562672492)
(('``', 'reviewername'), 0.0056502406232778754)
(('reviewername', "''"), 0.0056502406232778754)
(('.', '``'), 0.005033622557233089)
(('.', 'i'), 0.00407927293172189)
(('5.0', ','), 0.003884331263623606)
((':', '5.0'), 0.003884331263623606)
(('0', ','), 0.0038773591009591464)
(('[', '0'), 0.0038773591009591464)
((',', '0'), 0.0035822971769992226)
(('0', ']'), 0.0035820182904926475)
((',', '2013'), 0.0025214129059750877)
(('2013', "''"), 0.0025183451544027254)
(('.', 'the'), 0.0019307312850420895)
((',', 'and'), 0.0019112092295816034)
(('of', 'the'), 0.001832005461713345)
```

```
(('\\', "'"), 0.0018297743696607178)
(('.', 'it'), 0.001797144648391048)
(('``', 'i'), 0.001783758096075286)
((',', 'but'), 0.0017343951844109135)
(('2014', "'"), 0.0016083384834374881)
((',', '2014'), 0.0016077807104243314)
(('on', 'the'), 0.0014831184419837977)
(('in', 'the'), 0.0014778195983588087)
(('i', 'have'), 0.0013051888507867946)
```

9. This is the list of the top 50 words after removing the non alphabetical tokens and the st opwords.

```
(('works', 'great'), 0.0003547436363676929)
(('great', 'product'), 0.00021055931246667306)
(('works', 'well'), 0.00020024051172327318)
 (('make', 'sure'), 0.00017737181818384645)
(('well', 'made'), 0.00016593747141413308)
(('highly', 'recommend'), 0.00014976205403258733)
(('meguiar', "'s"), 0.00014864650800627382)
(("'ve", 'used'), 0.00013330775014446322)
(('would', 'recommend'), 0.00013274997713130646)
(('much', 'better'), 0.00012633558748000384)
(('battery', 'tender'), 0.00012187340337474984)
(('good', 'quality'), 0.00012159451686817146)
(('wiper', 'blades'), 0.00011322792167082022)
(('good', 'product'), 0.00011155460263134997)
(('griot', "'s"), 0.00011127571612477159)
(('wiper', 'blade'), 0.00010737137130503267434)
(('long', 'time'), 0.00010179357490110684)
(('high', 'quality'), 0.00010012025586163659)
(('work', 'well'), 9.900470983532309e-05)
(('car', 'wash'), 9.649473127611772e-05)
(('heavy', 'duty'), 9.649473127611772e-05)
(('work', 'great'), 8.980145511823672e-05)
(('oil', 'filter'), 8.952256861165834e-05)
(('first', 'time'), 8.589704402613947e-05)
(('great', 'price'), 8.422372498666922e-05)
(('car', "'s"), 8.394483848009085e-05)
(('worked', 'great'), 7.529935677616123e-05)
(('highly', 'recommended'), 7.00005131511721e-05)
(('looks', 'like'), 6.916385363143698e-05)
(('great', 'job'), 6.804830760512348e-05)
 (('good', 'price'), 6.721164808538835e-05)
(('travel', 'trailer'), 6.693276157880997e-05)
(('looks', 'great'), 6.665387507223161e-05)
 (('much', 'easier'), 6.386501000644786e-05)
(('clay', 'bar'), 6.135503144724249e-05)
(('pretty', 'good'), 6.079725843408573e-05)
(('microfiber', 'towels'), 5.996059891435061e-05)
(('perfect', 'fit'), 5.856166381458735e-05)
 (('amazon', 'customer'), 5.772950686172361e-05)
(('good', 'job'), 5.772950686172361e-05)
```

TRIVENI ASHOK NAIK
SUID: 406461644

10. The list of top 50 bigrams by their Mutual information scores using a minimum frequency of 5 is as given below

```
(('abraham', 'kovler'), 19.45189043547208)
(('afterwork', 'student'), 19.45189043547208)
(('angelo', 'mandato'), 19.45189043547208)
(('anthony', 'divenere'), 19.45189043547208)
(('artur', 'grabowski'), 19.45189043547208)
(('bono', 'publico\\'), 19.45189043547208)
(('bookie', 'monster\\'), 19.45189043547208)
(('borja', 'giralt'), 19.45189043547208)
(('bradley', 'olin'), 19.45189043547208)
(('darren', 'coyne'), 19.45189043547208)
(('demiko', 'dracket'), 19.45189043547208)
(('donovan', 'santos'), 19.45189043547208)
(('dyson', 'diva\\'), 19.45189043547208)
(('eclectic', 'reflectionz\\'), 19.45189043547208)
(('ernest', 'cheung'), 19.45189043547208)
(('fa', 'cabs'), 19.45189043547208)
(('fidel', 'medrano'), 19.45189043547208)
(('gato', 'flaco'), 19.45189043547208)
(('gib', 'sinep'), 19.45189043547208)
(('grenade', 'motorsports'), 19.45189043547208)
(('gsx', '1300\\'), 19.45189043547208)
(('hank', 'kramer'), 19.45189043547208)
(('hovey', 'corbin'), 19.45189043547208)
(('ian', 'macintyre'), 19.45189043547208)
(('jamesyn', 'quinn'), 19.45189043547208)
(('jimmie', 'lightner'), 19.45189043547208)
(('joanna', 'daneman'), 19.45189043547208)
(('katherine', 'laxague'), 19.45189043547208)
(('kilgore', 'gagarin'), 19.45189043547208)
(('leon', 'del'), 19.45189043547208)
(('linda', 'walchak'), 19.45189043547208)
(('mahindra', 'nagassar'), 19.45189043547208)
(('marilyn', 'koch'), 19.45189043547208)
(('mauricio', 'britva'), 19.45189043547208)
(('midtown', 'cop\\'), 19.45189043547208)
(('mihai', 'petre'), 19.45189043547208)
(('natalie', 'horschel'), 19.45189043547208)
(('natasha', 'chernavska'), 19.45189043547208)
(('neal', 'caffrey'), 19.45189043547208)
(('omm', 'noor'), 19.45189043547208)
(('papa', 'skittlz'), 19.45189043547208)
(('precession', 'guesswork\\'), 19.45189043547208)
(('reeve', 'lim'), 19.45189043547208)
(('robb', 'fladry'), 19.45189043547208)
(('rodney', 'lemke'), 19.45189043547208)
(('rudy', 'molina'), 19.45189043547208)
(('ryszard', 'sytnik'), 19.45189043547208)
(('sandra', 'ilgen'), 19.45189043547208)
(('sergeant', 'buzzfuzz'), 19.45189043547208)
```

```
(('shelley', 'gammon'), 19.45189043547208)
```

**Additional merit**

11. I have also computed the top 50 list of trigrams. The code and the output for this is given below.

```
automotivetrigrams = list(nltk.trigrams(automotivewords))
print(automotivetrigrams[:50])
```

```
[('{', '``', 'reviewerid'), ('``', 'reviewerid', "'"), ('reviewerid
', "'", ':'), ("'", ':', '``'), (':', '``', 'a3f73sc1ly51oo'), ('`
`', 'a3f73sc1ly51oo', "'"), ('a3f73sc1ly51oo', "'", ','), ("'", '
,', '``'), (',', '``', 'asin'), ('``', 'asin', "'"), ('asin', "'",
':'), ("'", ':', '``'), (':', '``', 'b00002243x'), ('``', 'b0000224
3x', "'"), ('b00002243x', "'", ','), ("'", ',', '``'), (',', '``'
, 'reviewername'), ('``', 'reviewername', "'"), ('reviewername', "'
'", ':'), ("'", ':', '``'), (':', '``', 'alan'), ('``', 'alan', 'mo
ntgomery'), ('alan', 'montgomery', "'"), ('montgomery', "'", ','),
("'", ',', '``'), (',', '``', 'helpful'), ('``', 'helpful', "'"),
('helpful', "'", ':'), ("'", ':', '['), (':', '[', '4'), ('[', '4'
, ','), ('4', ',', '4'), (',', '4', ']'), ('4', ']', ','), (']', ','
, '``'), (',', '``', 'reviewtext'), ('``', 'reviewtext', "'"), ('re
viewtext', "'", ':'), ("'", ':', '``'), (':', '``', 'i'), ('``', '
i', 'needed'), ('i', 'needed', 'a'), ('needed', 'a', 'set'), ('a', '
set', 'of'), ('set', 'of', 'jumper'), ('of', 'jumper', 'cables'), ('
jumper', 'cables', 'for'), ('cables', 'for', 'my'), ('for', 'my', 'n
ew'), ('my', 'new', 'car')]
```

b) For Musical Instruments

1. I have repeated the same things that I have done for Automotive Products to process the the reviews of Musical Instruments.
2. First I tokenized the text using the nltk.word_tokenize.
3. Then I found out the length of the words and it was 1869148
4. I found out the top 50 most frequent words. They are given below

```
``   165503
''   150264
,    139728
:    93563
.    48982
the         45545
i    31581
a    28977
and         27699
it   24838
to   23573
is   15250
```

```
for         14556
0   14025
of  13940
this        13039
overall     10528
you         10420
[   10359
]   10359
helpful     10328
summary     10273
asin        10264
}   10263
{   10262
reviewerid          10261
reviewtext          10261
unixreviewtime      10261
reviewtime          10261
reviewername        10234
that        9957
my  9819
with        9373
on  9363
in  8975
but         8316
have        7579
5.0         6938
!   6358
n't         6329
's  6326
not         6314
are         6088
guitar      6055
great       5940
as  5784
they        5056
good        4875
one         4570
so  4546
these       4545
was         4309
)   4304
very        4207
or  4168
be  4159
if  4108
do  4087
```

5. I applied a function to remove all the non-alphabetical words and also determine d the length of such words. The length of the words after removing the non alpha betical words was 1115463.

6. I used the below line to get all the words in lower case.
   musicwords = [w.lower() for w in musictokens]

The output is as follows.

```
['{', '``', 'reviewerid', "'", ':', '``', 'a2ibpi20uzir0u', "
'", ',', '``', 'asin', "'", ':', '``', '1384719342', "'", '
,', '``', 'reviewername', "'", ':', '``', 'cassandra', 'tu',
'\\', "'", 'yeah', ',', 'well', ',', 'that', "'s", 'just', 'l
ike', ',', 'u', '...', "'", ',', '``', 'helpful', "'", ':',
'[', '0', ',', '0', ']', ',', '``', 'reviewtext', "'", ':', '
``', 'not', 'much', 'to', 'write', 'about', 'here', ',', 'but'
, 'it', 'does', 'exactly', 'what', 'it', "'s", 'supposed', 'to
', '.', 'filters', 'out', 'the', 'pop', 'sounds', '.', 'now',
'my', 'recordings', 'are', 'much', 'more', 'crisp', '.', 'it',
'is', 'one', 'of', 'the', 'lowest', 'prices', 'pop', 'filters'
, 'on', 'amazon', 'so', 'might', 'as', 'well', 'buy', 'it', ',
', 'they', 'honestly', 'work', 'the', 'same', 'despite', 'thei
r']
```

## Additional merit

7. I created a new list of stopwords and added three extra stopwords to the existing steps.The following are the steps.

```
newstopwords = stopwords.append("can't, won't, n't, with")
print('number stopwords:', len(stopwords))
print(stopwords)
```

```
number stopwords: 157
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you'
, 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'h
imself', 'she', 'her', 'hers', 'herself', 'it', 'its', 'itself', 't
hey', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'wh
o', 'whom', 'this', 'that', 'these', 'those', 'am', 'is', 'are', 'w
as', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having',
'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if',
'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', '
with', 'about', 'against', 'between', 'into', 'through', 'during',
'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'i
n', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then'
, 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'a
ny', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such'
, 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', '
very', 's', 't', 'can', 'will', 'just', 'don', 'should', 'now', 'd'
, 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', 'couldn', 'didn',
'doesn', 'hadn', 'hasn', 'haven', 'isn', 'ma', 'mightn', 'mustn', '
needn', 'shan', 'shouldn', 'wasn', 'weren', 'won', 'wouldn', ‘can't
’, ‘won't ‘,‘n't’ ,’with’]
```

8. I have applied a function to remove the stopwords and also determined the lengt h of the document after removing the stopwords. The length of the document is 638803

9. The frequency distribution of top 50 words after removing the stopwords is given below

```
('overall', 10528)
('helpful', 10328)
('summary', 10273)
('asin', 10264)
('reviewerid', 10261)
('reviewtext', 10261)
('unixreviewtime', 10261)
('reviewtime', 10261)
('reviewername', 10234)
("'s", 6326)
('guitar', 6055)
('great', 5940)
('good', 4875)
('one', 4570)
('like', 4076)
('use', 3907)
('strings', 3778)
('sound', 3739)
('well', 3283)
('pedal', 2786)
('get', 2739)
('price', 2677)
('would', 2635)
('works', 2344)
('really', 2335)
('little', 2261)
('nice', 2223)
('amp', 2188)
("'ve", 2110)
('much', 2031)
('quality', 2001)
('better', 1805)
('used', 1796)
('also', 1727)
("'m", 1633)
('strap', 1619)
('easy', 1595)
('stand', 1579)
('tone', 1559)
('time', 1552)
('picks', 1481)
('using', 1452)
('bought', 1442)
('mic', 1412)
('need', 1379)
('tuner', 1376)
('best', 1353)
('product', 1349)
('work', 1340)
('even', 1323)
('play', 1309)
('buy', 1300)
```

```
('guitars', 1280)
('string', 1250)
('playing', 1220)
('cable', 1216)
('could', 1190)
('love', 1185)
('want', 1171)
('got', 1151)
('made', 1149)
('way', 1135)
('sounds', 1132)
('pedals', 1114)
('acoustic', 1109)
```

8. The list of the top 50 bigram frequenices is given below

```
(('helpful', 'reviewtext'), 0.009198870782804987)
(('overall', 'summary'), 0.009198870782804987)
(('unixreviewtime', 'reviewtime'), 0.009198870782804987)
(('reviewtime', 'reviewerid'), 0.009197974294082367)
(('it', "'s"), 0.003222876957819309)
(('reviewtext', 'i'), 0.003027442416288124)
(('of', 'the'), 0.0028768323108879453)
(('i', 'have'), 0.0025388560624601624)
(('on', 'the'), 0.0022223955433752 62)
(('if', 'you'), 0.0020673029943619825)
(('for', 'the'), 0.002061924062026262)
(('it', 'is'), 0.0020287539792893175)
(('is', 'a'), 0.0020188926033404965)
(('this', 'is'), 0.0018611105881593563)
(('in', 'the'), 0.0016898812421389145)
(('i', "'ve"), 0.00165491818195673)
(('and', 'the'), 0.0016127832119935848)
(('and', 'i'), 0.0015589938886363779)
(('for', 'a'), 0.001551821978855417)
(('with', 'the'), 0.0015392711367387354)
(('and', 'it'), 0.0014693450163743664)
(('to', 'the'), 0.001442450354695763)
(('i', "'m"), 0.001385971565170696)
(('reviewtext', 'this'), 0.0012281895499895558)
(('with', 'a'), 0.0012174316853181145)
(('summary', 'great'), 0.0011950194672526117)
(('but', 'i'), 0.0011717107604644888)
(('the', 'price'), 0.0011627458732382876)
(('to', 'be'), 0.0011206109032751421)
(('a', 'little'), 0.0011089565498810807)
(('i', 'was'), 0.0010838548656477176)
(('on', 'my'), 0.0010739934896988963)
(('you', 'can'), 0.0010641321137500 75)
(('have', 'a'), 0.0010488918054655332)
(('i', 'do'), 0.0010462023392976728)
(('to', 'use'), 0.0010211006550643097)
```

```
(('easy', 'to'), 0.0009942059933857061)
(('they', 'are'), 0.0009897235497726056)
(('that', 'i'), 0.0009717937753202033)
(('i', 'would'), 0.0009440026249189798)
(('but', 'it'), 0.0009287623166344379)
(('a', 'great'), 0.0009269693391891977)
(('it', 'does'), 0.0009072465872915551)
(('i', 'am'), 0.0009045571211236948)
(('so', 'i'), 0.0008821449030581920)
(('for', 'my'), 0.0008660081060510300)
(('in', 'a'), 0.0008525607752117282)
(('the', 'guitar'), 0.0008435958879855271)
(('a', 'good'), 0.0008426993992629070)
```

9. This is the list of the top 50 words after removing the non alphabetical tokens and the stopwords.

```
(('helpful', 'reviewtext'), 0.009198870782804987)
(('overall', 'summary'), 0.009198870782804987)
(('unixreviewtime', 'reviewtime'), 0.009198870782804987)
(('reviewtime', 'reviewerid'), 0.009197974294082367)
(('summary', 'great'), 0.0011950194672526117)
(('summary', 'good'), 0.0006625051660162641)
(('works', 'great'), 0.0004563127598136379)
(('price', 'unixreviewtime'), 0.0003594919777706656)
(('summary', 'nice'), 0.0003290113613012158176)
(('works', 'well'), 0.0003200464739753806 4)
(('summary', 'works'), 0.0002895658574062967 7)
(('acoustic', 'guitar'), 0.0002859799025158163)
(('great', 'unixreviewtime'), 0.0002626711957276933 5)
(('strings', 'unixreviewtime'), 0.0002617747070050732 4)
(('well', 'made'), 0.0002590852408372129)
(('planet', 'waves'), 0.0002563957746693525 7)
(('good', 'quality'), 0.0002277081355455089)
(('great', 'price'), 0.0002196397370419278 9)
(('would', "n't"), 0.0002133643159835870 7)
(('much', 'better'), 0.0002124678272609669 6)
(('electric', 'guitar'), 0.0002079853836478664)
(('great', 'product'), 0.0002061924062026261 8)
(('les', 'paul'), 0.0002035029400347658 4)
(('highly', 'recommend'), 0.0001999916985144285 4)
(('long', 'time'), 0.0001999916985144285 4)
(("'ve", 'used'), 0.0001954345415311848)
(('summary', 'excellent'), 0.0001873661430276038)
(('would', 'recommend'), 0.0001873661430276038)
 (('pedal', 'board'), 0.0001855731655823635 7)
(('summary', 'best'), 0.0001801942332466428 7)
(('reviewername', 'david'), 0.0001792977445240227 6)
 (('power', 'supply'), 0.0001757511789633542 3)
(('price', 'overall'), 0.0001739188121883020 8)
(('sound', 'great'), 0.0001694363685752015)
(('summary', 'perfect'), 0.0001694363685752015)
(('guitar', 'strings'), 0.0001640574362394808 2)
(('reviewtext', 'great'), 0.0001622644587942406)
(('product', 'unixreviewtime'), 0.0001586785039037601 5)
```

```
(('tuner', 'unixreviewtime'), 0.00015867850390376015)
(('high', 'quality'), 0.00015509254901327967)
(('sounds', 'great'), 0.00015419606029065956)
(('go', 'wrong'), 0.00015329957156803945)
(('ernie', 'ball'), 0.00015061010540017911)
(('gig', 'bag'), 0.00015061010540017911)
(('highly', 'recommended'), 0.0001488171279549389)
```

12. The list of top 50 bigrams by their Mutual information scores using a minimum frequency of 5 is as given below

```
(('abigail', 'ferreira'), 17.767283134498644)
(('airchamp', 'ariel\\'), 17.767283134498644)
(('alex', 'bartlett'), 17.767283134498644)
(('amk', 'tk\\'), 17.767283134498644)
(('andrea', 'polk'), 17.767283134498644)
(('ann', 'vande'), 17.767283134498644)
(('augusto', 'soto'), 17.767283134498644)
(('aurelio', 'abt\\'), 17.767283134498644)
(('barrelhouse', 'solly\\'), 17.767283134498644)
(('bertrarious', 'bertrarious\\'), 17.767283134498644)
(('bluesage', 'bluesage\\'), 17.767283134498644)
(('bnb', 'books\\'), 17.767283134498644)
(('brandfas', 'magicguitar\\'), 17.767283134498644)
(('brzozowski', 'tennessee\\'), 17.767283134498644)
(('bum-ki', 'cho'), 17.767283134498644)
(('caraballo', 'genrico\\'), 17.767283134498644)
(('caruso', 'gibsonjunkie\\'), 17.767283134498644)
(('casby', '-casby\\'), 17.767283134498644)
(('cedeno', 'jc\\'), 17.767283134498644)
(('cesar', 'augusto'), 17.767283134498644)
(('clare', 'chu'), 17.767283134498644)
(('coder10', 'ricardo\\'), 17.767283134498644)
(('corneto', 'beeteecee\\'), 17.767283134498644)
(('damien', 'esmond'), 17.767283134498644)
(('danyelle', 'mulin'), 17.767283134498644)
(('debra', 'merrill'), 17.767283134498644)
(('delos', 'dunn'), 17.767283134498644)
(('deux', 'bleus\\'), 17.767283134498644)

(('dias', 'jorge\\'), 17.767283134498644)
(('dm', 'guitarbabe\\'), 17.767283134498644)
(('docpain', 'docpain\\'), 17.767283134498644)
(('doreen', 'cascagnette'), 17.767283134498644)
(('e.i.e.i', 'owen'), 17.767283134498644)
(('federico', 'pacheco'), 17.767283134498644)
(('gerardo', 'piero'), 17.767283134498644)
(('irving', 'itis-truth.org\\'), 17.767283134498644)
(('jarrett', 'venturini'), 17.767283134498644)
(('jean', 'hanna'), 17.767283134498644)
(('jmac', 'jmac\\'), 17.767283134498644)
(('katheryn', 'bowling'), 17.767283134498644)
(('keane', "o'kelley"), 17.767283134498644)
(('labelle', 'nevermorefu\\'), 17.767283134498644)
```

```
(('lad', 'kbb\\'), 17.767283134498644)
(('li', 'amazon-aholic\\'), 17.767283134498644)
(('lochsa', 'lad'), 17.767283134498644)
(('louis', 'saad'), 17.767283134498644)
(('mattes', 'bill\\'), 17.767283134498644)
(('mccullar', 'peace\\'), 17.767283134498644)
(('md', 'zaheer'), 17.767283134498644)
(('mehrbach', 'zach\\'), 17.767283134498644)
```

13. This is the code that I used to get the trigrams.

```
musictrigrams = list(nltk.trigrams(musicwords))
print(musictrigrams[:50])
```

```
[('reviewerid', 'a2ibpi20uzir0u', 'asin'), ('a2ibpi20uzir0u', 'asin', 'rev
iewername'), ('asin', 'reviewername', 'cassandra'), ('reviewername', 'cass
andra', 'tu'), ('cassandra', 'tu', 'yeah'), ('tu', 'yeah', 'well'), ('yeah
', 'well', 'that'), ('well', 'that', "'s"), ('that', "'s", 'just'), ("'s",
'just', 'like'), ('just', 'like', 'u'), ('like', 'u', 'helpful'), ('u', 'h
elpful', 'reviewtext'), ('helpful', 'reviewtext', 'not'), ('reviewtext', '
not', 'much'), ('not', 'much', 'to'), ('much', 'to', 'write'), ('to', 'wri
te', 'about'), ('write', 'about', 'here'), ('about', 'here', 'but'), ('her
e', 'but', 'it'), ('but', 'it', 'does'), ('it', 'does', 'exactly'), ('does
', 'exactly', 'what'), ('exactly', 'what', 'it'), ('what', 'it', "'s"), ('
it', "'s", 'supposed'), ("'s", 'supposed', 'to'), ('supposed', 'to', 'filt
ers'), ('to', 'filters', 'out'), ('filters', 'out', 'the'), ('out', 'the',
'pop'), ('the', 'pop', 'sounds'), ('pop', 'sounds', 'now'), ('sounds', 'no
w', 'my'), ('now', 'my', 'recordings'), ('my', 'recordings', 'are'), ('rec
ordings', 'are', 'much'), ('are', 'much', 'more'), ('much', 'more', 'crisp
'), ('more', 'crisp', 'it'), ('crisp', 'it', 'is'), ('it', 'is', 'one'), (
'is', 'one', 'of'), ('one', 'of', 'the'), ('of', 'the', 'lowest'), ('the',
'lowest', 'prices'), ('lowest', 'prices', 'pop'), ('prices', 'pop', 'filte
rs'), ('pop', 'filters', 'on')]
```

**a. (10 pts) Description of processing steps: tokenization, lower case, stopwords or lemmatization, word frequencies, bigram frequencies and bigram PMI with frequency filter of 5 or greater, and state why you chose those options.**

### Task 2a) Briefly stating why the following processing is done:

- **Tokenizing -** Basic step so that It would be easy for further processing.
- **Lowering the case** – This is required to do because it helps in further processing and keeps the all the text in standard format.
- **Created a frequency of top 50 distribution of words**. – to get to know the frequency of the words, how many times the non – alphabetical, stopwords occur. This would help in further processing.

- **Applied the function to remove all the non-alphabetical words** – so that the comparision would be easy.
- **Applied a function to remove all the stopwords from the list** – to make it more clear and to get the words that could make sense and have some meaning.
- **Obtained the frequency distribution of top 50 words after removing all the stopwords from the list** – to get to know the frequency of the words that convey actual meaning.
- **Obtained the top 50 bigrams scores and their frequency**. – To get to know how many times each of the thing that is present is always being used.
- **Ran Pointwise Mutual Information on to the bigram list** – to find out the difference between the raw frequency score and the PMI Score frequency.
- **Apply a frequency filter of 5** – Helps us know which is the most common word that is repeated in the document.

**b. (10 pts) Discuss any issues with the lists and describe how the bigrams scored by frequency are different that the bigrams scored by PMI.**

## Task2b) Problems faced with the bigrams list

Some of the words here do not have real meaning
Eg : (('lad', 'kbb\\') , 17.767283134498644)

Non alphabetical characters still appear in the bigram list
(('amk', 'tk\\') , 17.767283134498644)

Some very long and weird bigrams are present.
(('**********09/27/2013still', 'lovin'), 20.089211229386006)

The **difference** between the top 50 bigrams by frequency and from the top 50 bigrams scored by Mutual Information are

**For Automotive Products**

- The scores that are obtained by frequency is between 0 – 10 and the scores that are obtained by the Pointwise Mutual Information starts from 19.0 for the Reviews of the Automotives.
- Bigrams using raw frequency score

- `(('wiper', 'blades'), 0.00011322792167082022)`
- `(('battery', 'tender'), 0.00012187340337474984)`
- `(('car', 'wash'), 9.649473127611772e-05)`
- `(('oil', 'filter'), 8.952256861165834e-05)`

- Bigrams using the Pointwise Mutual Information score

- (',power', 'stering'), 21.773818530359442)
- ('*edit', '3/29/14'), 21.773818530359442)
- ('***caution****seat', 'operation.-'), 21.773818530359442)
- ('-snap-on', '870116'), 21.773818530359442)

As you can see the PMI bigrams have digital data, hyphens, star marks where as the raw frequency bigrams do not have digital data.

**For Musical Instruments**

- The scores that are obtained by frequency is zero and the scores that are obtained by the Pointwise Mutual Information starts from 17.0 for the Reviews of the Musical Instruments.
- Bigrams using the Raw Frequency score

- ('acoustic', 'guitar'), 0.0002859799025158163)
- ('works', 'great'), 0.0004563127598136379)
- ('electric', 'guitar'), 0.0002079853836478664)
- ('sounds', 'great'), 0.00015419606029065956)

- Bigrams using the Pointwise Mutual Information score

- ("'electrified", "'it"), 20.089211229386006)
- ("'guitar", 'professor-qwik'), 20.089211229386006)
- ('150ms', '300ms'), 20.089211229386006)
- ('/lighter', 'tripod-style'), 20.089211229386006)
- ('16bit', '44khz'), 20.089211229386006)

As you can see the PMI bigrams have digital data where as the raw frequency bigrams do not have digital data.

- When applied PMI frequency with a filter of 5, the results are those of the names of the customer in the review list.

- ('abigail', 'ferreira'), 17.767283134498644)
- ('airchamp', 'ariel\\'), 17.767283134498644)

**3a. (10 pts) Define a comparison question between the two documents.**

TRIVENI ASHOK NAIK
SUID: 406461644

**(20 pts) Answer the question by picking examples from the lists and discussing how they show that the documents are different, not just reporting numbers. Discussion may include collection steps if significant, which will count towards discussion of differences.**

## Task 3 – Answering the question

The comparison question that I would like to compare here is

***"What kind of reviews is it? Which category of products is involved in the reviews?***

***"What people feel about the products? Good or Bad?"***

Musical Instruments:

After looking at the bigram frequencies we can easily find what kind of products are involved in the discussion or for what is the review.

(('the', 'guitar'))
(('electric', 'guitar'))
(('sounds', 'great'),
(('guitar', 'strings')
(('works', 'well'),
`'.poorly', 'executed.i'`

The usage of words like 'guitar', 'strings', 'sounds' and all tell that these are something related t o Musical Instruments.

Automotive Products :

(('battery', 'tender'),
(('wiper', 'blades'),
(('oil', 'filter'),
(('jumper', 'cables', 'for')
(('-', 'no', 'complaints')

The usage of words like the 'battery', 'blades', 'filter' ,'cables' tell that they are related to automoti ve products.

Even if the person does not know about the product, then by just looking at the reviews one can get to know about the product.

“Comparison about the reviews – like the Musical Instruments have some units like Hz, ms are present. The customers are talking how effective is the instrument and hence have given a detailed about the frequency of the instruments. But in Automotive Products reviews nothing of such kind is present.

```
(('120hz', '235hz'), 20.089211229386006)
('16gb', 'sandisc'), 20.089211229386006)
(('150ms', '300ms'), 20.089211229386006)
```

The second thing that I noticed in this product review is that it also, tells about what the customer felt about the products which tells us a lot about the product. Some of the bigrams show this. These were chosen from the bigram list – which tell about how good they felt about the product.

For Automotive Products:

```
('works', 'well')
('highly', 'recommend')
('-', 'no', 'complaints')
```

For Musical Instruments

```
('.poorly', 'executed.i')
(('would', 'recommend'),
```

There are also some similarities that I found between the two group of reviews.
The other comparision that I found is that age group of people who are providing the reviews. Some of the bigrams show that the people are within 30 years of age due to their words usage. These are some of the trigrams that are picked from both the documents showing similarity.

```
'acoustic'
'now', 'my', 'recordings'
'my', 'new', 'car'
```

Conclusion : The final Conclusion of this document is that it helps us determine for what kind of products is the reviews given by just looking at the bigrams.