Chirag Shah
19200072

Text Analytics COMP47600
Practical Number 4

13/10/2019

Q1) Find 10 short text-items (20-30 words); they could be emails, short docs, tweets or whatever… Make sure they all deal with some common topic of interest; so they have some of the same words

Solution: For 10 short text-items, the chosen topic is “Chernobyl” and the data is taken from multiple blogs namely -

TF-IDF, which stands for **term frequency inverse-document frequency**, is a statistic tha measures how important a term is relative to a document and to a **corpus**.

*TF-IDF(term) = TF(term in a document) * IDF(term)*

- *TF(term) = # of times the term appears in document / total # of terms in document*
- *IDF(term) = log(total # of documents / # of documents with term in it)*

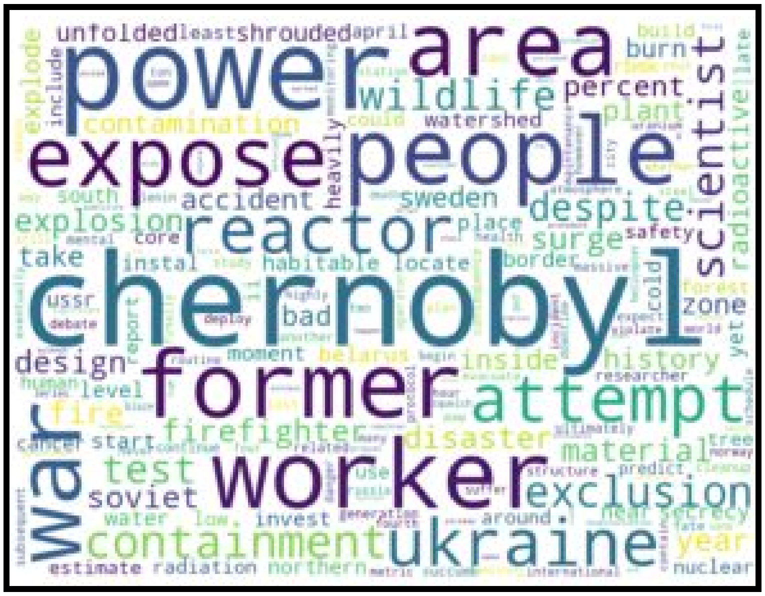
a.) Remove the standard stopwords from them using some standard list, use nltk.

→ The resultant list after stop words is:-

```
[['april', 'bad', 'nuclear', 'accident', 'chernobyl', 'history', 'unfolded', 'northern', 'ukraine', 'reactor', 'nucle
ar', 'power', 'plant', 'explode', 'burn', 'shrouded', 'secrecy', 'incident', 'watershed', 'moment', 'cold', 'war', 'h
istory', 'nuclear', 'power', 'chernobyl', 'year', 'scientist', 'estimate', 'zone', 'around', 'former', 'plant', 'habi
table', 'year', 'disaster', 'take', 'place', 'near', 'city', 'chernobyl', 'former', 'ussr', 'invest', 'heavily', 'nuc
lear', 'power', 'chernobyl', 'world', 'war', 'ii', 'start', 'chernobyl', 'soviet', 'scientist', 'instal', 'four', 'rb
mk', 'nuclear', 'reactor', 'power', 'plant', 'chernobyl', 'locate', 'south', 'ukraine', 'border', 'belarus'], ['april
', 'routine', 'maintenance', 'schedule', 'v', 'lenin', 'nuclear', 'power', 'station', 'fourth', 'reactor', 'worker',
'plan', 'use', 'downtime', 'test', 'whether', 'reactor', 'could', 'still', 'cool', 'plant', 'chernobyl', 'lose', 'pow
er', 'test', 'however', 'worker', 'violate', 'safety', 'protocol', 'chernobyl', 'power', 'surge', 'inside', 'plant',
'despite', 'attempt', 'chernobyl', 'shut', 'reactor', 'entirely', 'another', 'power', 'surge', 'cause', 'chain', 'rea
ction', 'explosion', 'inside', 'finally', 'nuclear', 'core', 'expose', 'chernobyl', 'spew', 'radioactive', 'material
', 'atmosphere', 'chernobyl', 'firefighter', 'attempt', 'put', 'series', 'blaze', 'plant', 'eventually', 'helicopter
', 'dump', 'sand', 'material', 'attempt', 'squelch', 'fire', 'contain', 'contamination', 'despite', 'death', 'two', '
people', 'chernobyl', 'explosion', 'hospitalization', 'worker', 'firefighter', 'danger', 'fallout', 'fire', 'one', 's
urround', 'chernobyl', 'area', 'include', 'nearby', 'city', 'pripyat', 'build', 'house', 'worker', 'chernobyl', 'plan
t', 'evacuate', 'hour', 'disaster', 'chernobyl', 'begin'], ['publicize', 'nuclear', 'accident', 'consider', 'chernoby
l', 'significant', 'political', 'risk', 'late', 'chernobyl', 'meltdown', 'already', 'spread', 'radiation', 'far', 'sw
eden', 'official', 'chernobyl', 'another', 'nuclear', 'plant', 'begin', 'ask', 'chernobyl', 'happen', 'ussr', 'first
', 'deny', 'accident', 'soviet', 'finally', 'make', 'brief', 'announcement', 'april', 'soon', 'world', 'realize', 'wi
tness', 'historic', 'event', 'percent', 'chernobyl', 'metric', 'ton', 'uranium', 'chernobyl', 'atmosphere', 'soviet',
'union', 'eventually', 'evacuate', 'people', 'establish', 'chernobyl', 'mile', 'wide', 'exclusion', 'zone', 'around',
'reactor', 'chernobyl', 'least', 'people', 'initially', 'die', 'result', 'accident', 'injure', 'united', 'nation', 's
cientific', 'committee', 'effect', 'atomic', 'radiation', 'chernobyl', 'report', 'child', 'adolescent', 'develop', 't
hyroid', 'cancer', 'expose', 'radiation', 'incident', 'although', 'expert', 'challenge', 'claim'], ['international',
```

b.) Compute the TF scores for all the remaining words in the texts and use R to show the word-cloud for these words. In your answer provide the matrix of TF scores and the word-cloud Image.

→ The resultant word cloud image is -



→ The snippet of matrix obtained is-

	abnormally	absence	absorb	accelerate	accident	action	active	\
Doc1	0	0	0	0	1	0	0	
Doc2	0	0	0	0	0	0	0	
Doc3	0	0	0	0	3	0	0	
Doc4	0	0	0	0	1	0	0	
Doc5	0	1	1	0	1	0	1	
Doc6	0	0	0	0	1	0	0	
Doc7	0	0	0	1	1	1	0	
Doc8	0	0	0	0	2	0	0	
Doc9	0	0	0	0	2	0	0	
Doc10	1	0	0	0	1	0	0	

	activity	adequate	admit	...	without	witness	wolves	worker	\
Doc1	0	0	0	...	0	0	0	0	
Doc2	0	0	0	...	0	0	0	4	
Doc3	0	0	0	...	0	1	0	0	
Doc4	0	0	0	...	0	0	0	0	
Doc5	1	0	0	...	0	0	1	0	
Doc6	0	0	0	...	0	0	0	0	
Doc7	0	1	0	...	2	0	0	0	
Doc8	0	0	0	...	0	0	0	0	
Doc9	0	0	0	...	0	0	0	0	
Doc10	0	0	1	...	0	0	0	1	

	world	worst	yat	year	yet	zone
Doc1	1	0	0	2	0	1
Doc2	0	0	0	0	0	0
Doc3	1	0	0	0	0	1
Doc4	0	0	0	0	1	0
Doc5	0	0	0	1	2	2
Doc6	0	0	0	0	0	0
Doc7	0	0	0	0	0	0
Doc8	0	0	0	0	0	0
Doc9	0	1	0	2	0	0
Doc10	0	0	1	1	0	0

[10 rows x 453 columns]

→ From the word cloud we can infer that the more prominent words have the following term frequency in each document.

df[‘reactor’] →

```
Doc1      2
Doc2      3
Doc3      1
Doc4      1
Doc5      0
Doc6      5
Doc7      7
Doc8      1
Doc9      0
Doc10     2
Name: reactor, dtype: int64
```

df[‘chernobyl’] →

```
Doc1      6
Doc2      9
Doc3      9
Doc4      5
Doc5      5
Doc6      6
Doc7      6
Doc8      4
Doc9      3
Doc10     5
Name: chernobyl, dtype: int64
```

df[‘power’] →

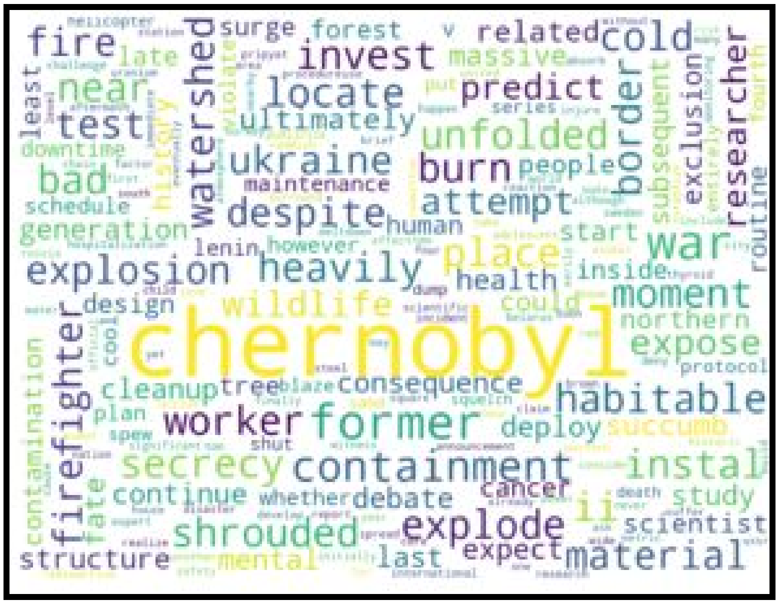
```
Doc1      4
Doc2      4
Doc3      0
Doc4      0
Doc5      1
Doc6      4
Doc7      7
Doc8      0
Doc9      0
Doc10     0
Name: power, d
```

c.) Now, compute the TF-IDF scores for all the same words in the texts. Construct a set of words that represents the TF-IDF scores you have found, for all the words. Use R to show a word-cloud for these words. Also, provide the matrix of TF-IDF scores and the word-cloud image.
Solution:

Following is the set of words that represent the TF-IDF score:-

```
[('area', 0.009120357638946648), ('people', 0.010670134423899063), ('accident', 0.012139252307821551), ('april', 0.01
2905648065943767), ('disaster', 0.012905648065943767), ('soviet', 0.013758024785639414), ('radioactive', 0.0144554248
50896181), ('reactor', 0.014790530899003052), ('nuclear', 0.01595094219921883), ('expose', 0.016483723269675895), ('c
ontain', 0.016483723269675895), ('evacuate', 0.016483723269675895), ('chernobyl', 0.018836608574568252), ('use', 0.01
9293908799913516), ('still', 0.019293908799913516), ('lose', 0.019293908799913516), ('safety', 0.019293908799913516),
('another', 0.019293908799913516), ('core', 0.019293908799913516), ('atmosphere', 0.019293908799913516), ('contaminat
ion', 0.019293908799913516), ('danger', 0.019293908799913516), ('include', 0.019293908799913516), ('build', 0.0192939
08799913516), ('begin', 0.019293908799913516), ('around', 0.020868707510887794), ('power', 0.021340268847798126), ('e
stimate', 0.023374729546129996), ('rbmk', 0.023374729546129996), ('city', 0.023575782489842826), ('station', 0.023575
782489842826), ('could', 0.023575782489842826), ('cause', 0.023575782489842826), ('chain', 0.023575782489842826), ('r
eaction', 0.023575782489842826), ('finally', 0.023575782489842826), ('eventually', 0.023575782489842826), ('two', 0.0
23575782489842826), ('fallout', 0.023575782489842826), ('surround', 0.023575782489842826), ('nearby', 0.023575782489
842826), ('hour', 0.023575782489842826), ('zone', 0.026654531244582298), ('take', 0.026654531244582298), ('ussr', 0.02
6654531244582298), ('belarus', 0.026654531244582298), ('plant', 0.03002867676145331), ('four', 0.031198661038158024),
('south', 0.031198661038158024), ('routine', 0.031551253589452245), ('maintenance', 0.031551253589452245), ('schedule
', 0.031551253589452245), ('v', 0.031551253589452245), ('lenin', 0.031551253589452245), ('fourth', 0.0315512535894522
45), ('plan', 0.031551253589452245), ('downtime', 0.031551253589452245), ('whether', 0.031551253589452245), ('cool',
0.031551253589452245), ('however', 0.031551253589452245), ('violate', 0.031551253589452245), ('protocol', 0.031551253
589452245), ('shut', 0.031551253589452245), ('entirely', 0.031551253589452245), ('spew', 0.031551253589452245), ('put
', 0.031551253589452245), ('series', 0.031551253589452245), ('blaze', 0.031551253589452245), ('helicopter', 0.0315512
53589452245), ('dump', 0.031551253589452245), ('sand', 0.031551253589452245), ('squelch', 0.031551253589452245), ('de
ath', 0.031551253589452245), ('hospitalization', 0.031551253589452245), ('one', 0.031551253589452245), ('pripyat', 0.
031551253589452245), ('house', 0.031551253589452245), ('surge', 0.03296744653935179), ('northern', 0.038122541898469
25), ('incident', 0.03812254189846925), ('world', 0.03812254189846925), ('start', 0.03812254189846925), ('inside', 0.0
3858781759982703), ('year', 0.04173741502177559), ('scientist', 0.04674945909225999), ('test', 0.04715156497968565),
('despite', 0.04715156497968565), ('explosion', 0.04715156497968565), ('material', 0.04715156497968565), ('firefighte
r', 0.04715156497968565), ('fire', 0.04715156497968565), ('attempt', 0.04945116980902768), ('bad', 0.0510190483574121
4), ('unfolded', 0.05101904835741214), ('explode', 0.05101904835741214), ('burn', 0.05101904835741214), ('shrouded',
0.05101904835741214), ('secrecy', 0.05101904835741214), ('watershed', 0.05101904835741214), ('moment', 0.051019048357
41214), ('cold', 0.05101904835741214), ('habitable', 0.05101904835741214), ('place', 0.05101904835741214), ('near',
0.05101904835741214), ('invest', 0.05101904835741214), ('heavily', 0.05101904835741214), ('ii', 0.05101904835741214),
('instal', 0.05101904835741214), ('locate', 0.05101904835741214), ('border', 0.05101904835741214), ('worker', 0.05782
1699403584725), ('history', 0.06239732207631605), ('ukraine', 0.06239732207631605), ('war', 0.0762450837969385), ('fo
rmer', 0.0762450837969385)]
```

Following is the generated word cloud image for TF-IDF-



Following is the generated matrix for the TF-IDF -

	terms	weights
0	area	0.009120
1	people	0.010670
2	accident	0.012139
3	april	0.012906
4	disaster	0.012906
5	soviet	0.013758
6	radioactive	0.014455
7	reactor	0.014791
8	nuclear	0.015951
9	expose	0.016484
10	contain	0.016484
11	evacuate	0.016484
12	chernobyl	0.018837
13	use	0.019294
14	still	0.019294
15	lose	0.019294
16	safety	0.019294
17	another	0.019294
18	core	0.019294
19	atmosphere	0.019294
20	contamination	0.019294
21	danger	0.019294
22	include	0.019294
23	build	0.019294
24	begin	0.019294
25	around	0.020869
26	power	0.021340
27	[illegible]	[illegible]

Q2) Using Python or R, compute the PMI scores for all adjacent pairs of words in your 10 - doc corpus (ie the texts after stop-word removal). List the top- 10 pairs based on the PMI scores found for the pairs. Do the results make sense? If not, then introduce a minimal cut - off frequency and re - compute the top - 10 until they seem sensible.

PMI → scoring 'ngrams' as per its mutual information. PMI is the correlation measure for two events considering specific events, *x* and *y.* Mutual information measures the PMI over all possible events: that is, MI is the average of PMI over all possible outcomes.

The formula for finding pmi score is -

$$\operatorname{pmi}(x;y) \equiv \log \frac{p(x,y)}{p(x)p(y)} = \log \frac{p(x|y)}{p(x)} = \log \frac{p(y|x)}{p(y)}.$$

The top 10 bigrams based on PMI scores found are as follows -

```
(('action', 'violation'), 9.712527000439824)
(('active', 'research'), 9.712527000439824)
(('adolescent', 'develop'), 9.712527000439824)
(('agricultural', 'land'), 9.712527000439824)
(('albeit', 'great'), 9.712527000439824)
(('along', 'mountainous'), 9.712527000439824)
(('although', 'expert'), 9.712527000439824)
(('animal', 'wake'), 9.712527000439824)
(('beckons', 'tourist'), 9.712527000439824)
(('belarusapril', 'routine'), 9.712527000439824)
(('beyond', 'legacy'), 9.712527000439824)
(('billion', 'damage'), 9.712527000439824)
(('bodø', 'along'), 9.712527000439824)
(('bolshomoshchnosty', 'kanalny'), 9.71252700043982
(('border', 'belarusapril'), 9.712527000439824)
(('brief', 'announcement'), 9.712527000439824)
(('budget', 'deal'), 9.712527000439824)
(('burn', 'shrouded'), 9.712527000439824)
(('bury', 'temporary'), 9.712527000439824)
```

The results don't make a great sense. For example - bury and temporary both the words have a PMI of 9.17. This is a known problem with PMI that it over-estimates the pointwise information of these adjacent words even when their frequencies is not very good.

To generate more sensible results, we introduce the minimal cut-off frequencies by setting the parameter apply_freq_filter. We pass in the value 2 to apply_freq_filter which in turn implies that only those bigrams that have a frequencies >=2 will be considered in the PMI score calculation process.
Following is the result generated after minimal cutoff frequency is set to 2. This result looks better and sensible.

```
(('chain', 'reaction'), 8.712527000439824)
(('metric', 'ton'), 8.712527000439824)
(('ton', 'uranium'), 8.712527000439824)
(('many', 'tree'), 8.127564499718668)
(('yet', 'full'), 8.127564499718668)
(('surge', 'cause'), 7.712527000439824)
(('exclusion', 'zone'), 7.29748950116098)
(('norway', 'sweden'), 7.127564499718668)
(('high', 'level'), 6.975561406273617)
(('soviet', 'union'), 6.390598905552461)
(('zone', 'around'), 6.127564499718667)
(('evacuate', 'people'), 5.90517207838222)
(('people', 'expose'), 5.90517207838222)
(('outside', 'soviet'), 5.805636404831305)
(('level', 'radiation'), 5.5832439834948575)
(('accident', 'include'), 5.542601998997512)
(('power', 'surge'), 5.390598905552462)
(('area', 'around'), 4.805636404831305)
(('reactor', 'core'), 4.6681328810813705)
(('reactor', 'use'), 4.6681328810813705)
(('low', 'power'), 4.390598905552462)
(('nuclear', 'power'), 4.127564499718667)
(('rbmk', 'reactor'), 3.931167286915164)
(('today', 'chernobyl'), 3.8796369862750817)
(('nuclear', 'accident'), 3.542601998997511)
(('chernobyl', 'report'), 2.8796369862750817)
(('power', 'plant'), 2.6901591874113695)
(('accident', 'chernobyl'), 2.2946744855539265)
(('chernobyl', 'area'), 1.557708891387719)
(('chernobyl', 'plant'), 1.17919726813399)
(('plant', 'chernobyl'), 1.17919726813399)
(('chernobyl', 'power'), 1.1426713921088751)
(('power', 'chernobyl'), 1.1426713921088751)
(('chernobyl', 'reactor'), 0.4202053676377844)
```

Q3) Entropy has been used to determine whether tweet set is interesting (contains variety) or repetitive (spam) Create two sets of 10 made-up tweets:
spam-set : where the 10 tweets are very similar containing an advert for a product
random-set : where the 10 tweets are very different, chosen at random from Twitter.
Now, find a python program or package that computes entropy and find the entropy values for (i) spam-set, (ii) random-set, (iii) the two sets combined

→ Entropy is usually defined as : **The Degree of Randomness**. The idea of entropy is to quantify the uncertainty of the probability distribution with respect to the possible classification classes. In general, it is a measure of disorder in the sense that all systems tend to, on their own, become less ordered.

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

Chirag Shah **Text Analytics COMP47600** **13/10/2019**
19200072 **Practical Number 4**

Contents of spam set :-

```
#Q3
spam_set = '''Crusts... love 'em or leave 'em?  RT for eat the crust LIKE for leave the crust #DominosPizza
Ready to rake in the dough? RT for a chance to #WinDominosPizza #DominosPizza
"People who put pineapple on pizza are the reason I have trust issues." #DominosPizza
Pizza = the best midnight snack. Hands down. #DominosPizza
Beauty comes in all different shapes and sizes: - Small Medium Large Hand Tossed Handmade Pan #DominosPizza
If you don't order pizza and watch scary movies, is it even #FridayThe13th #DominosPizza
Rock, paper, scissors for the last slice. It's the only fair way. #DominosPizza
Find someone you love seeing more than the Domino's delivery driver. #DominosPizza
A pizza slice just wants to feel whole again. You know it's real when you let them have the last slice. #DominosPizz
Life is short. Order the extra toppings. #DominosPizza
Surround yourself with people who say "let's order Domino's." #DominosPizza
Enjoys long walks to the fridge for leftover pizza. #DominosPizza'
'''
```

Contents of random set :-

```
random_set = '''
Four MIT graduates have just opened a restaurant where a robotic kitchen is preparing the meals
If developers were linguists... Person 1: How do you say "????" in Italian? Person 2: Why don't you use Japanese? It's
Peace of mind comes from accepting that you can't control what you can't control. Confidence comes from recognizing th
Tests should be coupled to the behavior of code and decoupled from the structure of code. Seeing tests that fail on bo
Today is the day I will officially never be on any 30 under 30 list
Never a dull moment with @sirajraval, who this time seems to have copied huge chunks of a paper, not even changing equ
live in Berkeley, not planning to leave but not super strongly attached either: if I have a reason to move, I would
Law of refactoring: Any small or large refactoring always leaves around bugs, it is only a matter of time till yo find
This AI software can predict and fill missing pieces in photos
Take a look at the writing process behind The Fountains of Silence in our exclusive interview with Goodreads Choice Aw
'''
```

Code snippet for the entropy is taken from - (NLTK Entropy)

```
import math
def entropy(labels):
    freqdst = nltk.FreqDist(labels)
    #print(freqdst)
    probs = [freqdst.freq(l) for l in freqdst]
    #print(probs)
    return -sum(p * math.log(p, 2) for p in probs)

print(entropy(word_tokenize(str(x))))
print(entropy(word_tokenize(str(y))))
print(entropy(str(x)+str(y)))
```

```
3.587157678834545
3.812991814263909
4.134252159057951
```

Here, entropy function has :-
Freqdst generates the frequency distribution
Probs list calculates the list of probability distributions
And then the standard entropy formula is returned by taking the sum and log of each probability generated for the words in the afore-mentioned list.
X - spam set
Y - random set
X + Y - combined set
From the result produced it is clearly observed that the entropy for spam set is less than the entropy for random set. The entropy for random set clearly means how randomly the data is varying.

References:-

NLTK Entropy and Information Gain, Link - http://www.nltk.org/book/ch06.html#fig-entropy