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

My text is some comments about the Truman Show:

```
text=['amazing movie watch explained person daily life scheduled another man thing superb interesting',
      'movie great definitely top movies time list amazing movie',
      'amazing movie everybody wondered life show everyone watching well',
      'absolutely terrific jim carrey legend one best movies ever seen watch help wonder reality heading direction',
      'definitely top movies time incredible moving movie love much read plot',
      'long time seen amazing fantasy movie unique story jim carrey performance outstanding',
      'finding truly meaning life amazing film one top favorite films',
      'amazing concept beautifully drafted creatively shot one favourites',
      'movie great could never expect move good hope everyone happy life',
      'movie really great great experience watching kind movie like really loved']
```

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

I used re.sub() to pre-process the text and used nltk.word_tokenize() to create the tokens of the text. Then I normalise the text to change the capital letters. Last I use the stop_words = set(stopwords.words('english')) to remove stopwords. Here is the outcome:

```
Out[161]: ' amazing movie watch explained person daily life scheduled
another man thing superb interesting movie great definitely top movies
time list amazing movie amazing movie everybody wondered life show
everyone watching well absolutely terrific jim carrey legend one best
movies ever seen watch help wonder reality heading direction definitely
top movies time incredible moving movie love much read plot long time
seen amazing fantasy movie unique story jim carrey performance
outstanding finding truly meaning life amazing film one top favorite
films amazing concept beautifully drafted creatively shot one favourites
movie great could never expect move good hope everyone happy life movie
really great great experience watching kind movie like really loved'
```

Q1(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 matix of TF scores and the word-cloud image.

TF scores:

Out[163]:

```
{'amazing': 0.05454545454545454,
 'movie': 0.08181818181818182,
 'watch': 0.01818181818181818,
 'explained': 0.00909090909090909,
 'person': 0.00909090909090909,
 'daily': 0.00909090909090909,
 'life': 0.03636363636363636,
 'scheduled': 0.00909090909090909,
 'another': 0.00909090909090909,
 'man': 0.00909090909090909,
 'thing': 0.00909090909090909,
 'superb': 0.00909090909090909,
 'interesting': 0.00909090909090909,
 'great': 0.03636363636363636,
 'definitely': 0.01818181818181818,
 'top': 0.02727272727272727,
 'movies': 0.02727272727272727,
 'time': 0.02727272727272727,
 'list': 0.00909090909090909,
 'everybody': 0.00909090909090909,
 'wondered': 0.00909090909090909,
 'show': 0.00909090909090909,
 'everyone': 0.01818181818181818,
 'watching': 0.01818181818181818,
 'well': 0.00909090909090909,
 'absolutely': 0.00909090909090909,
 'terrific': 0.00909090909090909,
 'jim': 0.01818181818181818,
 'carrey': 0.01818181818181818,
 'legend': 0.00909090909090909,
 'one': 0.02727272727272727,
 'best': 0.00909090909090909,
 'ever': 0.00909090909090909,
 'seen': 0.01818181818181818,
 'help': 0.00909090909090909,
 'wonder': 0.00909090909090909,
 'reality': 0.00909090909090909,
 'heading': 0.00909090909090909,
 'direction': 0.00909090909090909,
 'incredible': 0.00909090909090909,
 'moving': 0.00909090909090909,
 'love': 0.00909090909090909,
 'much': 0.00909090909090909,
 'read': 0.00909090909090909,
 'plot': 0.00909090909090909,
 'long': 0.00909090909090909,
 'fantasy': 0.00909090909090909,
 'unique': 0.00909090909090909,
 'story': 0.00909090909090909,
 'performance': 0.00909090909090909,
 'outstanding': 0.00909090909090909,
 'finding': 0.00909090909090909,
 'truly': 0.00909090909090909,
 'meaning': 0.00909090909090909,
 'film': 0.00909090909090909,
 'favorite': 0.00909090909090909,
 'films': 0.00909090909090909,
 'concept': 0.00909090909090909,
 'beautifully': 0.00909090909090909,
 'drafted': 0.00909090909090909,
 'creatively': 0.00909090909090909,
 'shot': 0.00909090909090909,
 'favourites': 0.00909090909090909,
 'could': 0.00909090909090909,
 'never': 0.00909090909090909,
 'expect': 0.00909090909090909,
 'move': 0.00909090909090909,
 'good': 0.00909090909090909,
 'hope': 0.00909090909090909,
 'happy': 0.00909090909090909,
 'really': 0.01818181818181818,
 'experience': 0.00909090909090909,
 'kind': 0.00909090909090909,
 'like': 0.00909090909090909,
 'loved': 0.00909090909090909}
```

Wordcloud:

Q1(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.

TD-IDF is like:(too big to show completely)

0.000000000000000000e+00	1.642231085278730296e-01	3.059136985833408318e-01	0.000000000000000000e+00	0.000000000000000000e+00	0.000000000000000000e+00
0.000000000000000000e+00	2.360091743282335397e-01	0.000000000000000000e+00	0.000000000000000000e+00	0.000000000000000000e+00	0.000000000000000000e+00
0.000000000000000000e+00	2.120619453806710519e-01	0.000000000000000000e+00	0.000000000000000000e+00	0.000000000000000000e+00	0.000000000000000000e+00
2.672909474004693808e-01	0.000000000000000000e+00	0.000000000000000000e+00	0.000000000000000000e+00	2.672909474004693808e-01	2.272217249856111065e-01
0.000000000000000000e+00	0.000000000000000000e+00	0.000000000000000000e+00	0.000000000000000000e+00	0.000000000000000000e+00	0.000000000000000000e+00
0.000000000000000000e+00	1.765376153749081412e-01	0.000000000000000000e+00	0.000000000000000000e+00	0.000000000000000000e+00	2.795551610364078732e-01
0.000000000000000000e+00	1.918262382222622187e-01	0.000000000000000000e+00	0.000000000000000000e+00	0.000000000000000000e+00	0.000000000000000000e+00
0.000000000000000000e+00	2.052416283984576928e-01	0.000000000000000000e+00	3.823227206540383838e-01	0.000000000000000000e+00	0.000000000000000000e+00
0.000000000000000000e+00	0.000000000000000000e+00	0.000000000000000000e+00	0.000000000000000000e+00	0.000000000000000000e+00	0.000000000000000000e+00
0.000000000000000000e+00	0.000000000000000000e+00	0.000000000000000000e+00	0.000000000000000000e+00	0.000000000000000000e+00	0.000000000000000000e+00

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.

My code:

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

finder = BigramCollocationFinder.from_words(word_tokenize(text))
```

```
....
(('absolutely', 'terrific'), 6.78135971352466)
(('another', 'man'), 6.78135971352466)
(('beautifully', 'drafted'), 6.78135971352466)
(('concept', 'beautifully'), 6.78135971352466)
(('could', 'never'), 6.78135971352466)
(('creatively', 'shot'), 6.78135971352466)
(('drafted', 'creatively'), 6.78135971352466)
(('everybody', 'wondered'), 6.78135971352466)
(('expect', 'move'), 6.78135971352466)
(('explained', 'person'), 6.78135971352466)
(('favorite', 'films'), 6.78135971352466)
(('finding', 'truly'), 6.78135971352466)
(('good', 'hope'), 6.78135971352466)
(('heading', 'direction'), 6.78135971352466)
(('help', 'wonder'), 6.78135971352466)
(('incredible', 'moving'), 6.78135971352466)
(('love', 'much'), 6.78135971352466)
(('man', 'thing'), 6.78135971352466)
(('move', 'good'), 6.78135971352466)
(('much', 'read'), 6.78135971352466)
(('never', 'expect'), 6.78135971352466)
(('outstanding', 'finding'), 6.78135971352466)
(('performance', 'outstanding'), 6.78135971352466)
(('person', 'daily'), 6.78135971352466)
(('plot', 'long'), 6.78135971352466)
(('read', 'plot'), 6.78135971352466)
(('reality', 'heading'), 6.78135971352466)
(('scheduled', 'another'), 6.78135971352466)
(('superb', 'interesting'), 6.78135971352466)
(('thing', 'superb'), 6.78135971352466)
(('truly', 'meaning'), 6.78135971352466)
(('unique', 'story'), 6.78135971352466)
(('well', 'absolutely'), 6.78135971352466)
(('wonder', 'reality'), 6.78135971352466)
(('jim', 'carrey'), 5.781359713524661)
(('carrey', 'legend'), 5.78135971352466)
(('carrey', 'performance'), 5.78135971352466)
(('direction', 'definitely'), 5.78135971352466)
(('ever', 'seen'), 5.78135971352466)
(('everyone', 'happy'), 5.78135971352466)
(('experience', 'watching'), 5.78135971352466)
(('hope', 'everyone'), 5.78135971352466)
(('like', 'really'), 5.78135971352466)
(('really', 'loved'), 5.78135971352466)
(('show', 'everyone'), 5.78135971352466)
(('story', 'jim'), 5.78135971352466)
(('terrific', 'jim'), 5.78135971352466)
(('watch', 'explained'), 5.78135971352466)
(('watch', 'help'), 5.78135971352466)
```

```
(('watching', 'kind'), 5.78135971352466)
(('watching', 'well'), 5.78135971352466)
(('definitely', 'top'), 5.196397212803505)
(('best', 'movies'), 5.196397212803504)
(('film', 'one'), 5.196397212803504)
(('legend', 'one'), 5.196397212803504)
(('long', 'time'), 5.196397212803504)
(('movies', 'ever'), 5.196397212803504)
(('one', 'best'), 5.196397212803504)
(('one', 'favourites'), 5.196397212803504)
(('shot', 'one'), 5.196397212803504)
(('time', 'incredible'), 5.196397212803504)
(('time', 'list'), 5.196397212803504)
(('top', 'favorite'), 5.196397212803504)
(('daily', 'life'), 4.78135971352466)
(('everyone', 'watching'), 4.78135971352466)
(('great', 'could'), 4.78135971352466)
(('great', 'experience'), 4.78135971352466)
(('happy', 'life'), 4.78135971352466)
(('life', 'scheduled'), 4.78135971352466)
(('life', 'show'), 4.78135971352466)
(('meaning', 'life'), 4.78135971352466)
(('seen', 'watch'), 4.78135971352466)
(('wondered', 'life'), 4.78135971352466)
(('movies', 'time'), 4.611434712082348)
(('top', 'movies'), 4.611434712082348)
(('amazing', 'concept'), 4.196397212803504)
(('amazing', 'fantasy'), 4.196397212803504)
(('amazing', 'film'), 4.196397212803504)
(('films', 'amazing'), 4.196397212803504)
(('list', 'amazing'), 4.196397212803504)
(('time', 'seen'), 4.196397212803504)
(('great', 'definitely'), 3.78135971352466)
(('really', 'great'), 3.78135971352466)
(('fantasy', 'movie'), 3.6114347120823473)
(('favourites', 'movie'), 3.6114347120823473)
(('interesting', 'movie'), 3.6114347120823473)
(('kind', 'movie'), 3.6114347120823473)
(('movie', 'everybody'), 3.6114347120823473)
(('movie', 'like'), 3.6114347120823473)
(('movie', 'love'), 3.6114347120823473)
(('movie', 'unique'), 3.6114347120823473)
(('moving', 'movie'), 3.6114347120823473)
(('one', 'top'), 3.6114347120823473)
(('seen', 'amazing'), 3.196397212803504)
(('great', 'great'), 2.78135971352466)
(('movie', 'great'), 2.6114347120823487)
(('movie', 'really'), 2.611434712082348)
(('movie', 'watch'), 2.611434712082348)
(('amazing', 'movie'), 2.611434712082347)
(('life', 'amazing'), 2.196397212803503)
(('life', 'movie'), 1.6114347120823478)
(('movie', 'amazing'), 1.0264722113611908)
```

Top 10:

('absolutely', 'terrific'), ('another', 'man'), ('beautifully', 'drafted'),('concept', 'beautifully'),('could', 'never'),('creatively', 'shot'), ('drafted', 'creatively'),('everybody', 'wondered'), ('expect', 'move'),('explained', 'person'),

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:

a. spam-set: where the 10 tweets are very similar containing a advert for a product

```
text2=['Swisse Sleep tablets don\'t knock you out, but you get a great sleep and forget about everything',
       'Magnesium is great for your muscles and can promote restful sleep! Pick-up Swisse Magnesium tablets from Woodlake',
       'It either helped last night or was a coincidence. I\'m gonna try again tonight. Those Swisse sleep vitamins are great!',
       'Have a devilish sleep on #Halloween night? Check out the great CVS deals on Swisse Sleep',
       'Buy some Swisse sleeping pills, they knock you out for a great night sleep, or try downloading a rain app to fall asleep too',
       'Stilnox and the Great Australian Sporting Sleep - Swisse may not be involved',
       'Swisse sleep vitamins really help my insomnia. You might want to give them a try. I hope you have a positive day!',
       'Swisse \'sleep\' has been working for me :-) seems to help ease those nagging thoughts that prevent you getting to sleep',
       'Hopefully these will help me sleep tonight',
       'Contemplating getting some #swisse sleep multivitamins to help me sleep at night!']
```

b. random-set: where the 10 tweets are very different, chosen at random from Twitter.

```
text3=['Breda O'Brien: UCD to turn on canonisation of John Henry Newman',
       'The European Union is so good at standing by its members, I wish we could join!',
       'outh Korea, Thailand, China and America all united on one stage and so many more nations in the audience',
       'i would commit several crimes to be able to try just one of these miku pringles',
       'Angela Merkel with the face of Steve Bruce.',
       'Im addicted to basketball at this point',
       'During our adventures I tried to teach the babies how to swim.In the middle of the Atlantic Ocean',
       'Joker kept winning at the box office',
       'My Truman Show!All The World Is A Stage!All The Actors Are In Place!',
       'Here's a baby Owl after a bath',]
```

Now, find a Python/R program or package that computes entropy and find the entropy values for (i) spam-set, (ii) random-set, (iii) the two sets combined.

Report the program you used and its source, the tweet data and the entropy values found.

```
In [177]: entropy(text_2)
Out[177]: 6.479275187681791

In [178]: entropy(text_3)
Out[178]: 6.433480549896259

In [179]: entropy(text_4)
Out[179]: 7.228142984858899
```

Text_2 is spam-set. Test_3 is random-set. Text_4 is two sets combined.

We can see that the spam-set has higher entropy than random-set. And the combined has the most entropy.

The program I used is to create token for two sets relatively and calculate their entropy.