Text Analytics: Practical Four

Question 1:

For my data I used Hillary Clinton's tweets coming up to the last US presidential election.

Question 1 a)

I used the NLTK Python library to remove standard stop words. However, due to the irregular nature of "real world data" there were still many instances of noise in the documents which needed to be properly cleaned for a fair analysis of the data. Below is a small summary of that process I used.

```
extra_stop_words =  [":", "?", "//", "...", ",", "``", ".", "''", "@", "'s", "(",
")", "https", "#", "-H", "'d", "n't", "ca",
"'re", "cl…", "3", "http", "-h", '→', ".", "b..."]

for tweet in tokenised_tweets:
for word in tweet:
if word in nltk.corpus.stopwords.words('english'):
tweet.remove(word)

for tweet in tokenised_tweets:
for word in tweet:
if word in extra_stop_words:
tweet.remove(word)
```

Question 1 b)

I computed the TF scores for all words in the documents using basic Python (*lists, sets, sum() etc*.) rather than a library. Below is a small section of the words with their TF scores across the ten documents, I could not include the full list of words due to the size constraints of this assignment.

words	0	1	2	3	4	5	6	7	8	9	words	0	1	2	3	4	5	6	7	8	9
made	0	0	0	0	0	0	0	2	0	0	home	0	0	0	0	0	0	0	0	1	0
american	0	0	0	0	1	0	0	0	0	0	like	0	0	0	0	0	1	0	0	0	0
person	0	0	0	0	0	0	0	0	1	0	hillary	0	0	0	0	0	0	1	1	1	1
take	0	0	0	0	1	0	0	0	0	0	action	1	0	0	0	0	0	0	0	0	0
donald	1	0	0	1	1	0	0	0	0	1	want	0	0	0	0	0	1	4	0	1	0
country	0	0	0	0	0	0	3	0	0	0	eating	0	0	0	0	0	1	0	0	0	0
night's	0	0	0	0	0	0	0	1	0	0	turn	0	0	0	0	0	0	1	0	0	0
plan	1	0	0	0	0	0	0	0	0	0	racial	0	0	1	0	0	0	0	0	0	0
unclear	0	0	0	0	0	0	0	1	0	0	kind	0	0	0	0	0	0	0	0	1	0
better	1	0	0	0	0	0	0	0	0	0	ready	0	2	0	0	0	0	0	0	0	0
put	1	0	0	0	0	0	0	0	0	0	think	0	0	0	0	0	0	0	0	1	0
big	0	0	0	0	0	0	1	0	0	0	life	1	0	0	0	0	0	0	0	0	0
night	1	1	0	0	0	0	0	0	0	0	dad	0	0	0	0	0	0	0	0	0	1
i'm	0	0	0	0	0	0	1	0	0	1	vision	0	1	0	0	0	0	0	0	0	0
wasn't	0	0	0	0	0	0	0	1	0	0	single	0	0	0	1	0	0	0	0	0	0
glad	0	0	0	0	0	0	0	0	0	1	vote	0	1	0	1	1	1	0	0	0	0
pound	0	0	0	0	0	1	0	0	0	0	week	0	0	0	0	1	0	0	0	0	0

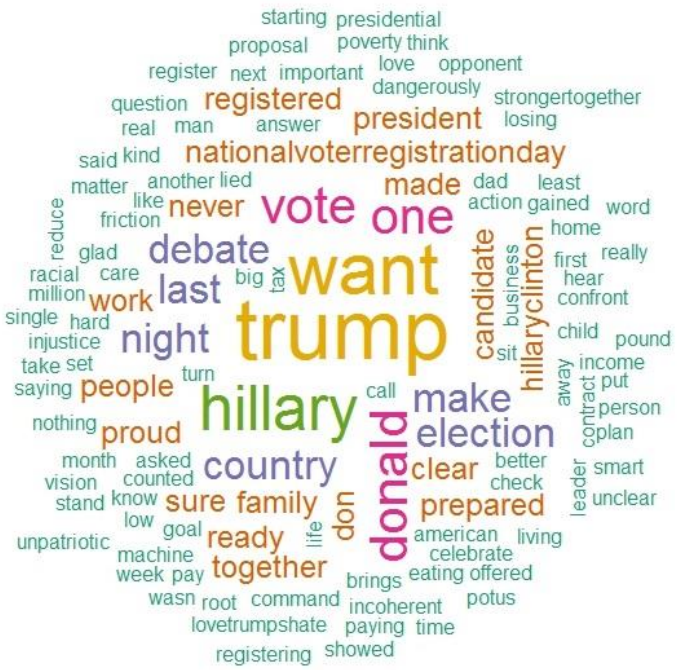

Figure 1 – R Word Cloud for question 1 b

Shown in figure 1 is my R word cloud plot with a *min.freq* = 1.

Question 1 c)

Below is a sample of my TF-IDF scores for the words in the corpus, again I could not include the complete list due to the size constraints of the assignment.

words	TF	IDF	0	1	2	3	4	5	6	7	8	9
made	2	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.40	0.00	0.00
american	1	10.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
person	1	10.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
poverty	1	10.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
take	1	10.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
time	1	10.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
losing	1	10.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
really	1	10.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
make	3	3.33	0.52	0.52	0.52	0.00	0.00	0.00	0.00	0.00	0.00	0.00
registered	2	5.00	0.00	0.00	0.70	0.00	0.70	0.00	0.00	0.00	0.00	0.00
prepared	2	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.40	0.00	0.00
could	1	10.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
first	1	10.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
work	2	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.70	0.00	0.00	0.70
tax	1	10.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
living	1	10.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
donald	4	2.50	0.40	0.00	0.00	0.40	0.40	0.00	0.00	0.00	0.00	0.40
country	3	3.33	0.00	0.00	0.00	0.00	0.00	0.00	1.57	0.00	0.00	0.00
night's	1	10.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00
plan	1	10.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

To construct a set of words which represents the TF-IDF scores I first totalled each word's TF-IDF scores across the documents into a new feature called *total-idf*, multiplied the values by four, rounded them and made a new paragraph represented by the frequencies of each word's new *total-idf* value. The code can be seen below and the results of the R word cloud in figure 2.

```
# Multiply the scores by 4 to get a frequency by which to represent them
df["total-idf"] = df["total-idf"]*4
# Conver to int to get rid of decimal places
df["total-idf"] = df["total-idf"].astype("int")

wordCloud = ""

for item, row in df.iterrows():
    wordCloud += (row["total-idf"])*(row.words + " ")
```

Question 2 a)

I used Python to calculate the PMI scores of every adjacent pair of words in the corpus. I again did not use a library and opted to do the calculations from scratch. Below is the final part of my calculations after counting the occurrences of words appropriately in conjunction with the frequency they occur together.

Figure 2 R Word Cloud for question 1 c.

```
# Calculating the PMI scores for each pair in the corpus
pmi_scores = list()
N = len(cw1w2)  # Count of each pair of words in whole corpus

for i in range(len(cw1w2)):
    pmi = math.log2(cw1w2[i]) + math.log2(N) - math.log2(w1c[i]) -
math.log2(w2c[i])
    pmi_scores.append(pmi)
```

After this I sorted the data by highest PMI scores to see which came up as most appropriate. Below (as requested) are the top 10 scores of my first attempt.

PMI(w1,w2)	c(w1)	c(w1,w2)	c(w2)	w1	w2
7.483816	1	1	1	eating	machine
7.483816	1	1	1	least	58
7.483816	1	1	1	leader	brings
7.483816	1	1	1	celebrate	registering
7.483816	1	1	1	like	eating
7.483816	1	1	1	55	pound
7.483816	1	1	1	gained	55
7.483816	1	1	1	first	presidential
7.483816	1	1	1	time	first
7.483816	1	1	1	58	time

These results do not really make sense, this is because PMI seriously over-estimates low frequency events because of how it treats counts, the solution is to introduce a minimum frequency cut-off. Below is when I only allow words with a minimum frequency of two.

PMI(w1,w2)	c(w1)	c(w1,w2)	c(w2)	w1	w2
6.483816	2	2	2	made	clear
5.898853	3	2	2	last	night
5.898853	3	2	2	make	sure
5.483816	2	1	2	registered	National…
5.483816	2	1	2	never	president
5.483816	2	1	2	candidate	made
5.483816	2	1	2	people	together
5.483816	2	1	2	president	National…
5.483816	2	1	2	hillaryclinton	ha
5.483816	2	1	2	i'm	proud

These results make more sense, finally I shall try a minimum frequency of three.

PMI(w1,w2)	c(w1)	c(w1,w2)	c(w2)	w1	w2
4.483816	4	3	6	donald	trump
3.483816	3	1	6	country	want
3.313891	4	1	4	hillary	one
2.898853	4	1	6	hillary	trump

These results make the most sense, but there is not enough data to have ten rows unfortunately. Nevertheless, I opted to include this table in this report because I feel it shows the power of PMI as the results begin to make more sense the larger the minimal cut-off frequency.

Question 3:

The tweet data I used for the question is as follows:

```
# Spam tweets advertising a product
spam_set = [
    "Hey, have you heard about our new fitness program called Greek God? If you
want to get shredded look no further.",
    "I used to be really skinny, then I got jacked and added 10lbs of muscle in
just 3 months, you can too!",
    "Want to be fit for life? The get my program called Greek God which will shred
you up in less than 3 months, money back guarantee",
    "Get fit on my new Greek God workout routine, if you don't like the results
you can get your money back after just 3 months, no risk!",
    "Are you tired of being skinny and out of shape? I was like that too but I got
in shape by following my Greek God routine",
    "This is not spam, I'm here to help you get into the best shape of your life,
my name is Greg and this is your lucky day.",
    "My new program Greek God will shred you up and your life will be as good as
mine, I promise you.... Greg",
```

```
    "Everyone! Have you heard of my new program Greek God? It will change your
life and you will be a sex symbol like me..."
    "I could never get girls until I got shredded on my new Greek God program.
Tired of a bad sex life? Get it now!",
    "Greg here, for black Friday there is a discount on my new Greek God program,
shred up and have sex with the girl of you dreams!",
    "If you're not happy with your body after 3 months I will give you a full
money back guarantee. But you will be, sex and being shredded is guaranteed!"
]

# Random tweets collected from Twitter
random_set = [
    "News just in, processed meat causes cancer and we are all screwed!",
    "Being vegan is one of the best thing you can do for your health, you can call
yourself an environmentalist if you don't!",
    "XBox live is only €3.95 a month, get online now and play with your friends!",
    "Hi Hillary could you please stop spamming Trump, I want America to get great
again! Go Trump!",
    "Michelle & I are praying for the victims in Las Vegas. Our thoughts are with
their families & everyone enduring another senseless tragedy.",
    "Happy bday Joseph! You’re a fantastic son & a great training partner. You get
stronger/smarter every year & I’m so proud of you. I love you.",
    "More and more people are suggesting that Republicans (and me) should be given
Equal Time on T.V. when you look at the one-sided coverage?",
    "Thank you to all who participated in today's discussion on opioid abuse. By
talking about it, we can start to make a real difference.",
    "Like that you’re puttin’ your goals out there in the universe. Now the fun
part - puttin’ in the work. Good luck dude",
    "Become a 'legend' never crossed my mind when I was selling polaroids of
myself for $5 at flea markets 20yrs ago. Very cool.. thank you. 🙌
https://twitter.com/legendmgz/status/911189045278994433 …"
]
```

To compute entropy for this question I used the program supplied by Mark Keane in the lecture notes for *Lecture 4. Beyond Frequencies.*

```
def entropy(labels):
    """Source is the lecture slides"""

    freqdist = nltk.FreqDist(labels)
    probs = [freqdist.freq(l) for l in freqdist]
    return -sum(p * math.log(p,2) for p in probs)
```

The entropy value for the spam set was 6.68, the entropy value for the Random set was 7.13 and the entropy value for the spam and random set combined was 7.56.

Entropy is the level of disorder in a sample, it makes sense that the spam set would have the lowest level of disorder as it is easier to predict by using much of the same language. The random set had higher entropy which also makes sense since it is completely random. Finally, the combined set had even higher entropy, this illustrates that even if a subset of data in a sample (half high entropy, half low) is somewhat predictable the overall entropy will likely still be worse than the part of the sample with higher entropy by itself.