**Text mining and qualitative research on transcripts of oral evidence given to the parliamentary inquiry entitled 'Social media data and real time analytics'**

DS7001 Coursework / Student No. 1720146

Chung Hoi Chiu

**Abstract:**

This coursework reports the following:

1. The processes and findings of using various text cleaning techniques and text mining techniques to clean and study the three transcripts of oral evidence given in the meetings held by the Science and Technology Committee (hereinafter referred to as ‘Committee’) of the House of Commons in 2014. The oral evidence is part of the contributions provided to the inquiry of the Committee entitled 'Social media data and real time analytics' (reference) (the transcripts are hereinafter referred to as 'Documents').

2. The results of analysing the Documents by using the Gioia methodology (reference) and the qualitative data analysis software RQDA.

Data files and code used for doing this coursework are located at https://github.com/ericchchiu/u1720146\_CN8001\_data

**1. Text cleaning of the Documents**

The procedures for changing the pdf format files of the Documents to plain text files, cleaning them and adjusting their layout are as follows:

a. Open the pdf format files with Adobe Acrobat Reader DC.

b. Choose File -> Save as Text to save each of them as a text file.

c. Open the text files with Notepad++.

d. For each text files, observe whether there are strange characters with black backgrounds. If the answer is positive, delete them by following the next step.

e. For example, if character pairs 'FF' with black background are found, they are the 'Form Feed' symbol, ASCII 12. They are left there after forms are rejected when converting a pdf format file to a text format file. They should be deleted by choosing Search -> Replace (or its corresponding icon). Then, in the window opened, select 'Regular expression' and then replace all '\0x12' with '' (Quotes should be ignored. So '' means blank).

f. Choose View -> Show Symbol -> Show All Characters. Then all unseem characters will be shown. Such characters include CR which is 'Carriage return' symbol, ASCII 13, regex escape syntex \r; and LF which is 'Line feed' symbol, ASCII 10, regex escape syntax \n. For deleting all blank lines, one should go to the Replace window mentioned above, and replace '\r\n\r\n' with '\r\n' (Quotes should be ignored). Since there are some double or even trible blank lines, the said replacement operation should be repeated until all blank lines are deleted.

g. Since the files will be analysed with computer, it is safe to convert all non-basic ASCII characters (that is, those characters not within the first 128 unicodes) to basic ASCII characters. The method is as follows: choose Search -> Find Characters in Range (at bottom) -> Non-ASCII Characters (128-255), then find and replace all non-basic ASCII characters with their corresponding basic ASCII characters. For example, all right single quotation mark 146 should be changed to single quote 39 and all pound sign should be changed to GBP .

(note: When coding, attention should be paid to some special characters even they are basic ASCII characters.

For example, in the following code, the special meaning of ' is escaped by the escape character \:

csvText[csvText$content=='D\'Mari', ]

h. The following two lines of R code can be used to find and show lines of a text file which containing non-basic ASCII code:

Data <- readLines("Data.csv")

tools::showNonASCII(Data)

i. The files should be treated as csv files with | as the delimiter. Therefore, Notepad++ regular expressions should be used to change expressions such as:

Q147 Professor van Zoonen: Yes, but an ombudsman is...

Q147 Dr d'Aquin: There is an element that relates to...

to:

Q147|Professor van Zoonen|Yes, but an ombudsman is...

Q147|Dr d'Aquin|There is an element that relates to...

Method: in the Replace window, type:

Find what: (^Q\d+)(\s)([A-Z,a-z,’]+\s\*[A-Z,a-z,']\*\s\*[A-Z,a-z,']\*\s\*)(:)(\s)

Replace with: \1|\3|

then click Replace All

(The correctness of the regular expressions shown above can be checked by copying and plasting the above first two lines of text to a new Notepad++ file and then perform the abovementioned operation.)

The same changes can also be done with the below R code:

tx2 <- gsub(pattern = "(^Q\\d+)(\\s)([A-Z,a-z,’]+\\s\*[A-Z,a-z,']\*\\s\*[A-Z,a-z,']\*\\s\*)(:)(\\s)", replace = "\\1|\\3|", tx)

j. Lines within a file which are not intended to be input to R are commented out with #. Lines which are intended to be crammed in one cell are wraped with ^.

Below is the content of a sample file csvText.txt:

#Sample text:

1|^"When I say 'immediately,' I mean some time before August," said the manager.

"Why did she call the man a 'traitor'?"^

2|^'Good morning, D'Mari,' called D'Juan.

"Good morning, D'Mari," called Jane.^

The code for reading csvText.txt into R, converting it to a dataframe and viewing it is as follows:

csvText <- read.csv('csvText.txt', comment.char = '#', quote = '^"\'', sep = '|', stringsAsFactors = FALSE)

View(csvText)

(note: ^ is added as an additional quote character)

k. The following five clean-up files produced by following the above-mentioned techniques are kept in https://github.com/ericchchiu/u1720146\_CN8001\_data

u1720146\_CN8001\_oral\_20140618.csv

u1720146\_CN8001\_oral\_20140623.csv

u1720146\_CN8001\_oral\_20140708.csv

u1720146\_CN8001\_Responsible\_Use\_of\_Data.csv

u1720146\_CN8001\_persons.csv

(The penultimate file is converted from the Committee’s report entitled ‘Responsible Use of Data (*reference*). The last file has three columns: person, sector and description. The person column includes all persons attended the meetings at which the oral evidence was provided. The description column consists brief descriptions of all persons which were extracted from the three oral evidence transcripts. The sector column assigns a sector to each person according to their descriptions. Sectors include member, academic, business, government and ngo.)

**2. Text mining of the Documents**

Code referred to in this section is located at Appendix 1 of this coursework.

Working directory was set and packages to be used were loaded to the RStudio at the beginning of text mining (please refer to code 01).

Data files were loaded to the RStudio and converted to dataframes (please refer to code 02). The dataframes were then copied to a database (pleased refer to code 03). A dataframe which containing utterances of all witnesses was then created by using tables in the database and RSQLite code (please refer to code 04).

**2.1 Production of a comparison word cloud and a commonality word cloud**

A comparison cloud and a commonality cloud in respect of the witnesses Emma Carr and Timo Hannay were produced and shown below (please refer to code 05):

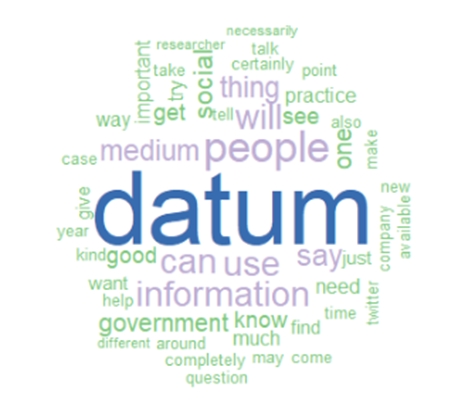
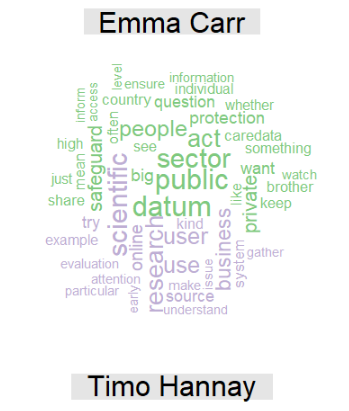


Figure 2.1 The comparison cloud Figure 2.2 The commonality cloud

(Emma Carr vs. Timo Hannay) (Emma Carr and Timo Hannay)

The reason for choosing the utterances of Emma Carr and Timo Hannay for comparison is that according to the information provided by the Documents, Timo Hannay was the only witness came from the private business sector and Emma Carr was the Acting Director of an NGO called Big Brother Watch, which, after conducting a mini desktop research, was found an organization very distant from business.

Content of the two clouds are illuminative. Emma Carr said private, public, sector and safeguard especially more frequently than Timo Hannay, while Timo Hannay said business, research, scientific especially more frequently than Emma Carr. They both mentioned datum, information and people frequently.

The biggest word in the commonality cloud is datum instead of data. The reason is that within the tm package function tm\_map, the textstem package function lemmatize\_strings is used to lemmatise words. The R code is reproduced below for easy of reference:

THAndECUttrnces\_cor\_cl <- tm\_map(THAndECUttrnces\_cor\_cl, lemmatize\_strings)

**2.2 Producing functions to produce word clouds and to perform word counts and word associations**

For avoiding repetition of code, before further analysing the Documents, functions were produced to perform the tasks as stated in the title of this sub-section 2.2 above (please refer to code 06).

Relatively important features about the functions are as follows:

a. Function 6.4 ranks the word frequency list first according to the frequencies (in descending order) of the words and then according to the alphabets of the words (in ascending order) and the relevant code is shown below:

sctorUttrnces\_wordCount[order(-sctorUttrnces\_wordCount[,2], sctorUttrnces\_wordCount[,1]),]

The abovementioned ranking method results in a list produced by Function 4 the same as the one produced by Function 6.5. Function 6.5 uses freq\_terms, a qdap package function, to word frequency list directly from corpus.

b. Function 6.6 uses findAssocs, a tm package function, to produce association word lists. By inputting a word or a character vector of more than one word to the function, association word lists in respect of the word or those words can be produced in one execution. The last parameter of the findAssocs function can also be a numeric or a numeric vector.

However, the output of the findAssocs function is a vector list. In order to make the output from Function 6.6 more readable, it can be feed into Function 6.7 to convert it to a dataframe. Function 6.7 uses list\_vect2df, a qdap package function, to convert a vector list to a dataframe.

**2.3 Applying the functions described in 2.2 to study the Documents**

Based on information provided by the Documents and a mini desktop research, the witnesses were divided into four sectors, namely, academic (10 persons), business (2), government (2) and NGO (2). Their utterances were accordingly divided into four groups. Answers to each question provided by a group were treated as a document and therefore concatenated (please refer to code 07).

The functions developed and mentioned in section 2.2 above were then used to analyse the texts production of which is described in the above paragraph (please refer to code 08).

**2.3.1 Production of word clouds for every of the four sector’s utterances**

The four word clouds are shown below:

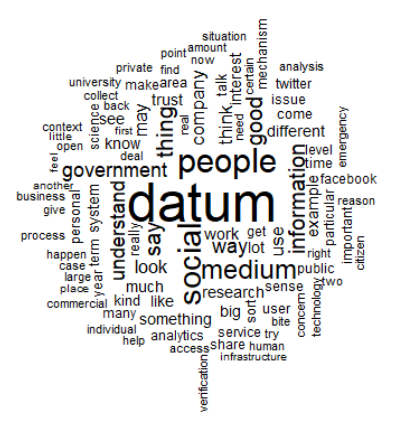
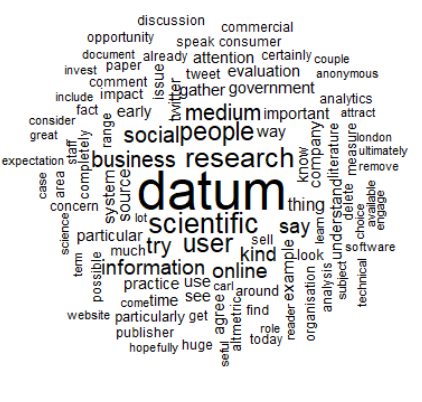
 

Figure 2.3 Academic sector word cloud Figure 2.4 Business sector word cloud

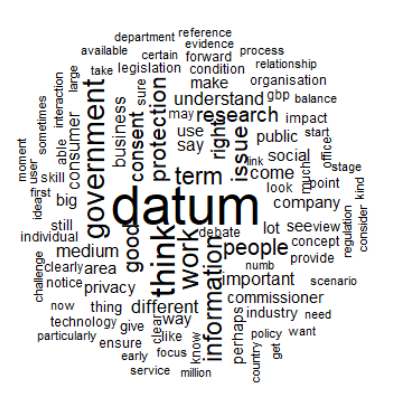
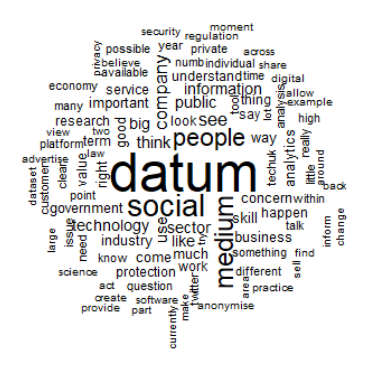
 

Figure 2.5 Government sector word cloud Figure 2.6 NGO sector word cloud

**2.3.2 Production of word frequency lists for every of the four sector’s utterances**

The csv files of word frequency lists produced by forming tdms (Function 6.3), summing rows of the tdms and then ranking the rows (Function 6.4) should be identical to the word frequency lists produced directly from corpus by using freq\_term, a qdap package function (Fuction 6.5). The two lines of Code 8.6 is for forming a dataframe of the word frequency lists of the sectors. It is shown below:



Figure 2.7 Dataframe of four word frequency lists (only the top 20 words are shown)

**2.3.3 Production of word associaton lists**

In code 8.7, Function 6.7 was used to find words associated with certain selected words of the four sectors. The words selected are datum, government, legislation and research. The degree of association for these four words are set as 0.4, 0.5, 0.5, .06 respectively.

In code 8.8, an output from execution of code 8.7, the association word lists in respect of government sector utterances, which is a vector, was feed into Function 6.8 to convert the lists to a dataframe to make the result more readable.

Part of the dataframe (rows 58 to 105) and the code for obtaining this part of the dataframe is shown below:

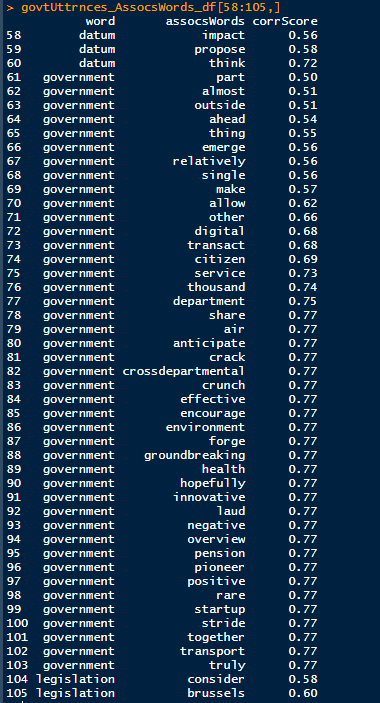


Figure 2.8 Part of a dataframe which is for showing words associated with ‘government’ and the correlation value equal or higher than 0.5 in the government utterances corpus

Code 09 is for plotting a graph from rows of the above dataframe (Figure 2.8) with the values of them in the ‘word’ column equals ‘government’. The graph is shown below:

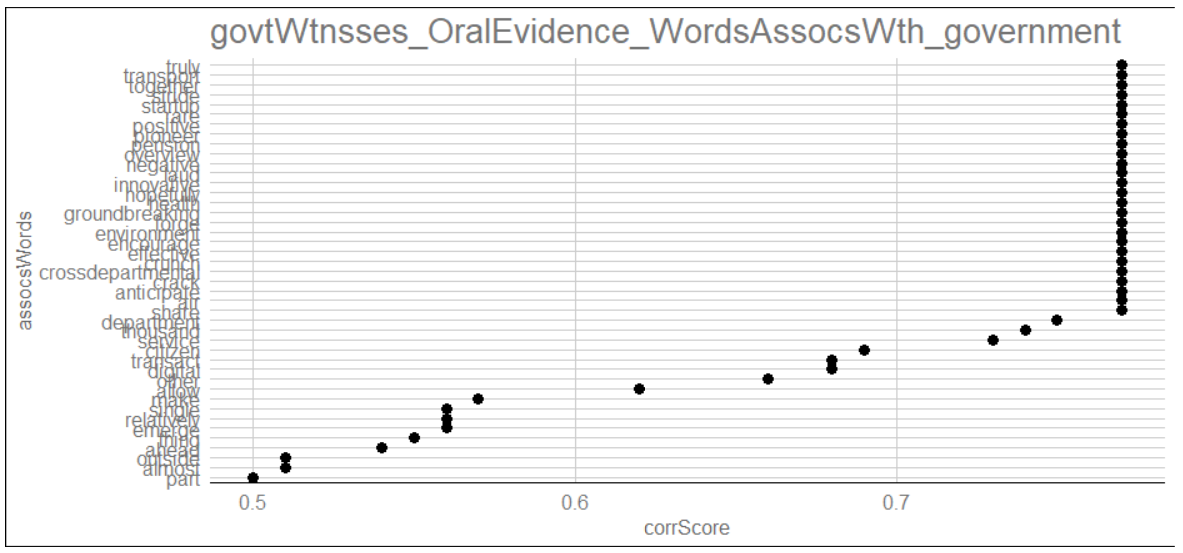


Figure 2.9 Graph plotted with values in the assocsWords column against values in the corrScore column of those rows of the dataframe shown in Figure 2.8 values in the word column of which equals ‘government’

2.3.4 Attempt of Application of the tf-idf technique to attribute each of the

