Earnings Calls Analysis Part 1

Dave Hurst

Sunday, November 11, 2014

The goal of this exercise is to explore the corpus of Earnings Calls transcripts using R. The corpus is made up of transcripts for the primary two hard disk drive manufactures, Seagate (STX) and Western Digital (WDC). The transcripts for each quarter were divided into two separate documents, one for the corporate report, and one for Q&A. These documents were gathered for the fical Q4 (ending in June) 2013, Q4 2014, and Q1 2015, totalling 12 documents in all.

Load R packages:

```
library(tm) # Framework for text mining.
library(SnowballC) # Provides wordStem() for stemming.
library(RColorBrewer) # Generate palette of colours for plots.
library(ggplot2) # Plot word frequencies.
library(magrittr)
#library(Rgraphviz) # Correlation plots  #no package available for 3.1.1 from cran
```

Corpus (text documents) are in a folder in the working directory named EarningsCalls

```
corpus_dir <- "EarningsCalls"
cname <- file.path(".", corpus_dir)
nfiles <- length(dir(cname))
```

There are 12 files in corpus directory (EarningsCalls) The text files are available for download here: https://github.com/dsdaveh/R453-Text-Analytics/tree/master/EarningsCalls

```
dir(corpus_dir)
```

```
##  [1] "STQ 2013Q4 Earnings Q&A.txt" "STQ 2013Q4 Earnings.txt"
##  [3] "STQ 2014Q4 Earnings Q&A.txt" "STQ 2014Q4 Earnings.txt"
##  [5] "STQ 2015Q1 Earnings Q&A.txt" "STQ 2015Q1 Earnings.txt"
##  [7] "WDC 2013Q4 Earnings Q&A.txt" "WDC 2013Q4 Earnings.txt"
##  [9] "WDC 2014Q4 Earnings Q&A.txt" "WDC 2014Q4 Earnings.txt"
## [11] "WDC 2015Q1 Earnings Q&A.txt" "WDC 2015Q1 Earnings.txt"
```

```
docs <- Corpus(DirSource(cname))
class(docs)
```

```
## [1] "VCorpus" "Corpus"
```

```
class(docs[[1]])
```

```
## [1] "PlainTextDocument" "TextDocument"
```

```
#inspect(docs[1])  #commented out since this output is large

toSpace <- content_transformer(function(x, pattern) gsub(pattern, " ", x))

# doc style specific transfomration
docs <- tm_map(docs, toSpace, "http:\\S*") #remove URLS
docs <- tm_map(docs, toSpace, "<.*/.*>") #remove page numbers
docs <- tm_map(docs, toSpace, "\\n") #remove newline chars
#TO-DO: remove unprintable characters

getTransformations() # this is the list of things we can do to the corpus
```

```
## [1] "removeNumbers"     "removePunctuation" "removeWords"
## [4] "stemDocument"      "stripWhitespace"
```

```
docs <- tm_map(docs, content_transformer(tolower))
docs <- tm_map(docs, content_transformer(removePunctuation))
docs <- tm_map(docs, removeWords, stopwords("english"))
docs <- tm_map(docs, removeWords, c("question")) #equivalent to SPSS concept delete
docs <- tm_map(docs, stripWhitespace)
docs <- tm_map(docs, removeNumbers)  #may want to reconsider doing this

#docs <- tm_map(docs, stemDocument)  #step words -- not sure we want to do this
#TO-DO: compare the differences of stemming and non stemming
```

Now the the data is prepared we can create a document term matrix and explore the frequencies. Heres the raw DTM.

```
dtm <- DocumentTermMatrix(docs)
dtm
```

```
## <<DocumentTermMatrix (documents: 12, terms: 3013)>>
## Non-/sparse entries: 8348/27808
## Sparsity           : 77%
## Maximal term length: 21
## Weighting          : term frequency (tf)
```

```
nterms <- dtm$ncol
inspect(dtm[1:5, c(12:14, 1000:1002, seq(nterms-2,nterms))])  #first 11 have unprintable chars
```

```
## <<DocumentTermMatrix (documents: 5, terms: 9)>>
## Non-/sparse entries: 16/29
## Sparsity           : 64%
## Maximal term length: 10
## Weighting          : term frequency (tf)
##
##                            Terms
## Docs                        aaron ability able faq far fast zero
##   STQ 2013Q4 Earnings Q&A.txt     6       3    3   0   2    0    0
##   STQ 2013Q4 Earnings.txt         0       0    0   0   0    0    0
##   STQ 2014Q4 Earnings Q&A.txt     7       1    4   0   1    0    0
```

```
##   STQ 2014Q4 Earnings.txt         0       0    0   0   1    0    0
##   STQ 2015Q1 Earnings Q&A.txt     5       0    3   0   5    1    1
##                              Terms
## Docs                          zettabyte zettabytes
##   STQ 2013Q4 Earnings Q&A.txt         0          0
##   STQ 2013Q4 Earnings.txt             0          0
##   STQ 2014Q4 Earnings Q&A.txt         2          1
##   STQ 2014Q4 Earnings.txt             0          0
##   STQ 2015Q1 Earnings Q&A.txt         0          0
```

To do a little more investigation, I created some simple functions:
- *mostFrequentTerms* - displays the counts for the most frequent terms in the corpus
- *fintTerms* - lists all the terms in the corpus that match a patter and their counts

```
mostFrequentTerms <- function (x.dtm, n=6) {
  freq <- colSums(as.matrix(x.dtm))
  ord <- order(freq)
  freq[tail(ord, n)]
}

mostFrequentTerms( dtm )
```

```
##   million president      just      will     think   quarter
##       134       136       163       167       225       247
```

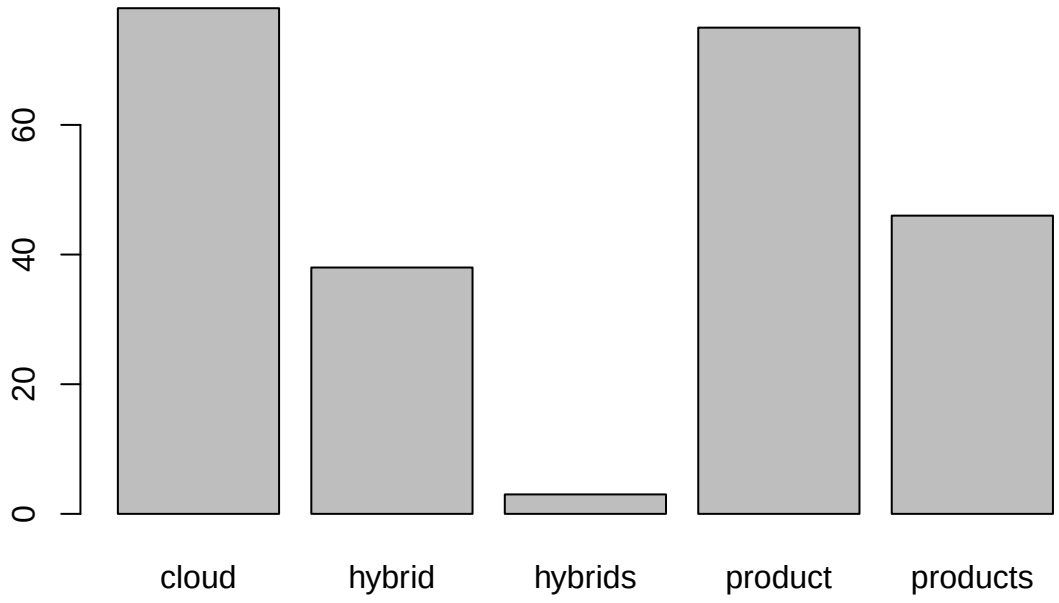
```
findTerms <- function (x.dtm, pattern) {
  freq <- colSums(as.matrix(x.dtm))
  terms <- names(freq)
  freq[grep(pattern, terms)]
}

findTerms( dtm, "product")
```

```
##    product production   products
##         75          7         46
```

An obvious improvement is that we probably want to count “product” and “products” as a single concept. Before we do, let’s plot some of the more domain relevant concepts. Note the need for some regex magic to cover the plurals.

```
barplot( findTerms(dtm, "^cloud$|^hybrids*$|^products*$") )
```



So far so good, but I'd like to merge some of the different firms, such as the product/products example above. I originally tried to create a list of stemmed words and then group all the words that matched that, but that strategy was too broad and would have ended up merging words like "act" and "cataract", so the manual oversight would have been overwhelming. Instead I settled for merging plurals and past tense: dress = dressed = dresses.

The approach I took was to sort all the terms and then use pattern matching to group neighboring words together into 'synonyms' (I probably should have called these concepts in my code to be consistent with SPSS). The sort was key in making this efficient, since there were 3000+ terms in the corpus (approximate number of patterns to check $= 3000 * 2999/2 = 4.5M$). I assumed one of the matches I was looking for could be found within 10 words, limiting the number of searches to approximately $10 * 3000 = 30K$. I wrote the match to a CSV file with the idea that I could use Excel to edit the list, and read it back in – kind of a poor man's resource template. I only modified a few of the concepts, as I just wanted to make sure it worked. Along the way I discovered I could play with things interactively (the beauty of R). Here's a little narrative to demonstrate:

The first step is to build a seed file with concepts based on plural/past tense word forms:

```
dtm.dict <- colnames(as.matrix(dtm))          #a list of all the terms
synonym.df <- data.frame( root=as.character(dtm.dict), stringsAsFactors=FALSE) #a processing frame

findWordForms <- function(v) {
   # v is a character vector
   # v[1] is the root used for the search
```

```
   # look in v for plural or past tense forms and return them as a vector (with root as first term)
   # -- use this with a sorted list and pass the next 10 (or so) words as part of the vector
   pattern <- sprintf("%se?d$|%se?s$", v[1], v[1])
   strings <- v[2:length(v)]
   c(v[1], strings[grep( pattern, strings)])
}

words <- list()
for(i in 1:(nrow(synonym.df))) {
  words[i] <- list( findWordForms( dtm.dict[i:(i+10)]) )
}
synonym.df$wordList <- words
synonym.df$n <- apply(synonym.df, 1, function(x) length(unlist(x$wordList)))  #count the number of syno
synonym.df <- subset(synonym.df, n>1)  #reduce to words with synonyms
synonym.df$csv <- apply(synonym.df, 1, function(x) paste(unlist(x$wordList), collapse=",")) #create a c

outfile <- "synonym_seed.csv"
infile <- "synonym.csv"

write.table( synonym.df[synonym.df$n > 1, ]$csv, file=outfile,
             quote=FALSE, row.names=FALSE, col.names=c("primary term"), sep=",")
rm(synonym.df)
```

The output of the above code extract is a file called synonym_seed.csv. This needs to be copied (by the user) and edited as appropriate. The manually edited file should be called synonym.csv. This prevents the manual edits from being overwritten by the program. In order for things to work properly, the .csv file should be saved using Excel.

```
synonyms <- read.csv(infile, stringsAsFactors=FALSE)
head(synonyms)
```

```
##   primary.term            X X.1
## 1   accelerate  accelerated
## 2       access     accessed
## 3  accommodate accommodates
## 4      account     accounts
## 5      accrual     accruals
## 6      achieve     achieved
```

Next I wrote a function to replace all the concepts with the first word in the group.

```
replaceSynonyms <- function(x, syn) {
  #replace synonyms with primary term
  syn
  for (i in 1:nrow(syn)) {
    #if(i%%100 == 0) print(sprintf("%d...", i))
    root <- syn[i,1]
    j <- 2
    while (!is.null(syn[i,j]) && grepl("[a-z]", syn[i,j]) ) {   #first entry that has no chars (most re
      pattern <- sprintf("%s", syn[i,j])
      x <- tm_map(x, content_transformer(function(y) gsub(pattern, root, y)))
      j <- j+1
```

```
    }
  }
  x
}

docs <- replaceSynonyms(docs, synonyms)
dtm <- DocumentTermMatrix(docs)

findTerms( dtm, "product") #for comparison post combining word forms
```

```
##    product production
##        121          7
```

```
mostFrequentTerms( dtm )
```

```
##    just    will    year  market   think quarter
##     163     167     168     190     225     293
```

year and *market* now roll up into the top 6.

I also noticed the word "think" is high on the list. Given the nature of the corpus (an earnings call), I suppose that "think" is really being used in the context of forecasting, so I thought it would be interesting to roll those terms up together. First lets look at all the candidates to combine into a single concept

```
psyn <- findTerms(dtm, "predict|forecast|think")
psyn
```

```
##    forecast forecasting     predict predictable  predicting  prediction
##           9           3           4           1           1           3
##       think    thinking
##         225          11
```

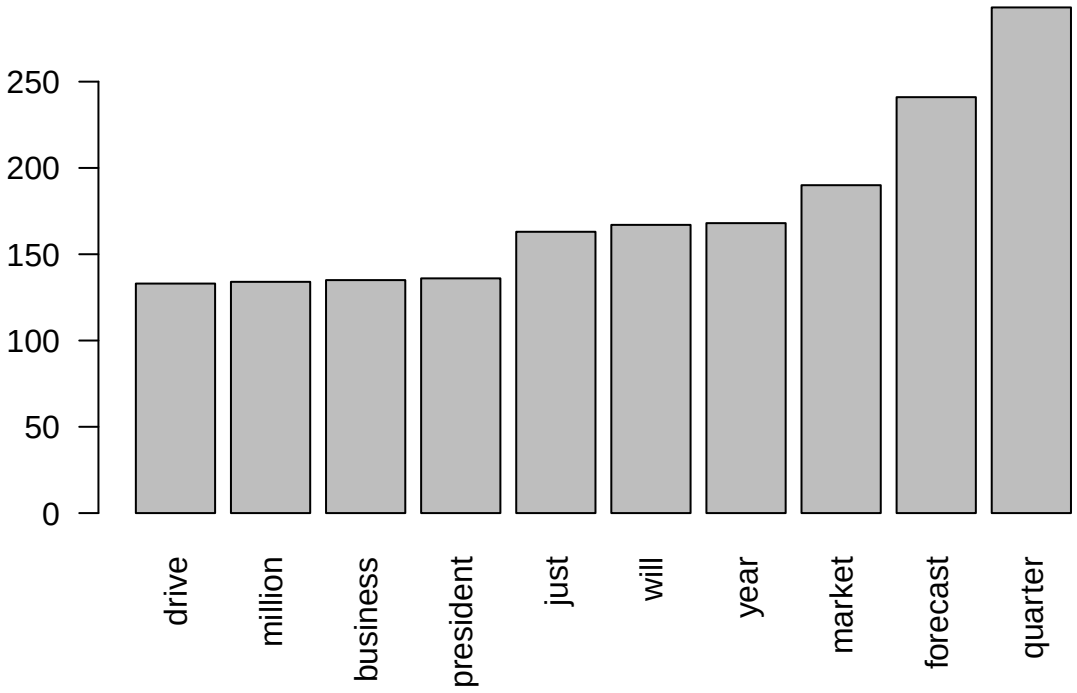
Its reasonable to lump all these together, with exception of "predictable" which is more of an adjective, so I remove that and replace any occurances of these words with "forecast", everything rolls up into that one word.

```
psyn.words <- names(psyn)[- grep("predictable", names(psyn))]  #remove "predictable"
docs <- replaceSynonyms(docs, data.frame(matrix( psyn.words, nrow=1 )))
dtm <- DocumentTermMatrix(docs)
mostFrequentTerms( dtm )
```

```
##     just     will     year   market forecast  quarter
##      163      167      168      190      241      293
```

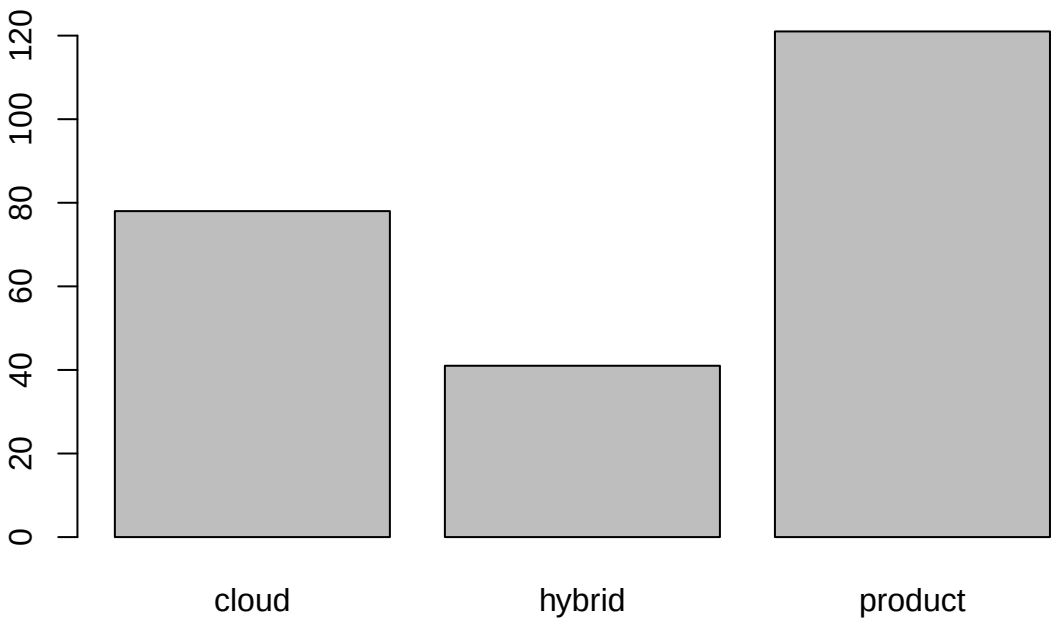
Notice "think" has been replaced by "forecast" in the top six, and gets a bit of a bump in the word count. Let's plot the top 10.

```
par(las=2)
barplot(mostFrequentTerms( dtm, 10 ))
```



and here is our relevant terms plot with the concepts rolled up.

```
par(las=0)
barplot(findTerms(dtm, "^cloud$|^hybrid|^product$"))  #word boundaries need work
```



And that's about all the fun I want to have with single word concepts. Time to move on to word pairs... (to be continued)

References: - http://handsondatascience.com/TextMiningO.pdf

The data and code used for this exercise can be found on Github at https://github.com/dsdaveh/R453-Text-Analytics

The most recent verson of this document is available here