LDA Topic Model Analysis of Facebook Data

Dr Timothy Graham (ANU)

5 June 2018

Introduction

In this tutorial we will be doing a small-scale project that uses an advanced text analysis technique (topic modelling) to study Breast Cancer Awareness on Facebook (similar to the journal article in the set reading). Note: this is tutorial has been adapted from undergraduate course work in the Bachelor of Advanced Computing at the ANU.

You will learn how to:

1. Import and clean text data from Facebook (user comments data);
2. Convert the data into a document-term matrix (DTM) and prepare it for topic modelling;
3. Generate a Latent-Dirichlet Allocation (LDA) topic model for the data;
4. Analyse and interpret the topics that are generated from the model to provide a qualitative analysis of the main topics or themes that are discussed within the Facebook data.

The dataset for this tutorial consists of 10,538 user comments that were posted on two pages within the timeframe of 15/3/2014 to 15/3/2017 (three years of data). The Facebook pages are:

1. Breast Cancer Support UK (https://www.facebook.com/breastcancersupportuk/)

2. Breast Cancer Awareness Month (https://www.facebook.com/breastcancerawarenessmonth/)

The dataset combines together user comments collected from both of these pages.

You can download the dataset and save it to your local working directory in R. It has been provided for you in the course materials folder.

What are topic models?

Informally, topic models can be said to 'automagically' extract topics from text. Topic models unveil hidden thematic structure in a collection of text documents. As Scott Weingart describes in his excellent blog post: "if I feed the computer, say, the last few speeches of President Barack Obama, it'll come back telling me that the president mainly talks about the economy, jobs, the Middle East, the upcoming election, and so forth".

More formally, topic models are "generative models which provide a probabilistic framework for the term frequency occurrences in documents in a given corpus" (Grun & Hornik, 2011, p. 1). For an overview of probabilistic topic models that balances ease-of-reading with technical detail, see Wood (2014).

Latent Dirichlet allocation (LDA) is a particularly popular method for fitting a topic model. It treats each document as a mixture of topics, and each topic as a mixture of words. This allows documents to "overlap" each other in terms of content, rather than being separated into discrete groups, in a way that mirrors typical use of natural language.

LDA topic models have become a very powerful method for unsupervised classification of text data.

Installation of R and RStudio

This tutorial will be conducted using the R programming language. You will need to download the R base package and optionally RStudio if you prefer a graphical user interface. Personally, I would encourage you to install RStudio as it makes project management much easier. It will also mean that you can open this tutorial RMarkdown file in RStudio and run the code directly.

Tip: Create a new *project* in RStudio where you can store your files for this project.

Once you have installed R (and optionally RStudio), you need to install several packages (libraries) for this tutorial.

Run the following code and it will install the required packages:

```r
install.packages("topicmodels")
install.packages("tm")
install.packages("slam")
install.packages("stringr")
install.packages("Rmpfr")
```

Load libraries and dataset

After installing R and the required packages, the next step is to load them into the R global environment:

```
## Loading required package: topicmodels
```

```
## Loading required package: tm
```

```
## Loading required package: NLP
```

```
## Loading required package: slam
```

```
## Loading required package: stringr
```

```
## Loading required package: Rmpfr
```

```
## Loading required package: gmp
```

```
## Error: package or namespace load failed for 'gmp':
##  package 'gmp' was installed by an R version with different internals; it needs to be reinstalled fo
```

Next, we will import the dataset. It might take a minute or two to download from the server…

```r
rawData <- read.csv("dataset_COMP2550_breast_cancer_facebook_comments.csv",
    stringsAsFactors = F)
# View(rawData)
```

Data cleaning / wrangling

We need to do some cleaning and wrangling of the data in order to make it ready for analysis.

```r
# convert the dataframe object to a character vector
rawData <- as.character(rawData[, 1])

# create a vector object of the word count for each comment
wordCounts <- str_count(rawData, "\\S+")
# find which comments have less than N words (in this example
# we specify 20 words)
toDel <- which(wordCounts < 20)
# remove these comments from the dataset
rawData <- rawData[-toDel]

# convert the character encoding to UTF-8 (to avoid problems
# with weird/strange characters in the data) we will
# duplicate the comments vector into a new vector
# 'facebookData' facebookData <-
# iconv(rawData,to='utf-8-mac') # <--- !! Only use this if
# you are working on a Mac computer
facebookData <- iconv(rawData, to = "utf-8")
```

Next we will convert the vector of comments to a VCorpus object to do text manipulation using the 'tm' package in R

```
facebookDataCorpus <- VCorpus(VectorSource(facebookData))
```

Now we will create a document-term matrix from the data, which we will be using as input for the topic model.

There are several choices that we make in the arguments of the `DocumentTermMatrix` function, which will affect the output of the topic model. For example, if we set `stopwords` to `T`, then this means we will remove common English words from the dataset such as 'the' and 'and' (which we don't think will be very useful for our analysis).

```
# create a document term matrix out of the Vcorpus object
dtmTopicModeling <- DocumentTermMatrix(facebookDataCorpus, control = list(stemming = F,
    tolower = TRUE, removeNumbers = F, removePunctuation = TRUE,
    language = "english", stopwords = T))
```

We can print some information to the console about the document term matrix. We are particularly interested in the number of terms (columns). For larger datasets it can become very computationally expensive to run topic models with large number of terms or features in the matrix.

```
dtmTopicModeling
```

```
## <<DocumentTermMatrix (documents: 3228, terms: 8459)>>
## Non-/sparse entries: 94644/27211008
## Sparsity           : 100%
## Maximal term length: 557
## Weighting          : term frequency (tf)
```

Sometimes there are URLs in the data which we want to remove. We can do this via:

```
dtmTopicModeling <- dtmTopicModeling[, !grepl("http", dtmTopicModeling$dimnames$Terms)]
```

The final part of data cleaning involves applying a technique called 'term frequency-inverse document frequency' (tf-idf). We will use the approach in Grun & Hornik (2011), which includes a formula on page 12. Tf-idf is a principled approach to filter out 'unimportant' words from our text. As Grun & Hornik describe: "this measure allows to omit terms which have low frequency as well as those occurring in many documents" (2011, p. 12). Words with a higher tf-idf score are more important. We will only keep words that have a tf-idf greater than or equal to the median tf-idf score of the sample.

We also use tf-idf to reduce the size of the document-term matrix `dtmTopicModeling` that we created in the previous step. Size is not really a problem for the little dataset in this tutorial, but as datasets become bigger the feature set (i.e., the number of columns in the document-term matrix) becomes computationally expensive to work with.

```
term_tfidf <- tapply(dtmTopicModeling$v/row_sums(dtmTopicModeling)[dtmTopicModeling$i],
    dtmTopicModeling$j, mean) * log2(nDocs(dtmTopicModeling)/col_sums(dtmTopicModeling >
    0))
median_tfidf <- summary(term_tfidf)[3]
dtmTopicModeling <- dtmTopicModeling[, term_tfidf >= median_tfidf]
```

Some final necessary cleaning of the data to ensure that there are no rows in the document-term matrix (comments) that have no text (terms), which can result from the tf-idf process and other processing.

```r
# find which rows no longer have any words in them (row sums
# to zero) due to tf-idf
toRemove <- which(row_sums(dtmTopicModeling) == 0, )
# ensure that our dtm only contains rows where sum > 0
dtmTopicModeling <- dtmTopicModeling[row_sums(dtmTopicModeling) >
    0, ]

dtmTopicModeling$dimnames$Docs <- as.character(c(1:dtmTopicModeling$nrow))  # fix the 'row' numbering
```

Generate the topic models

In this section we will be generating the topic model and analysing it.

```r
k <- 10  # the number of topics (you will need to choose a different number for your assignment!)

seedNum <- 1  # specify a seed number so we can replicate the results

# create the topic model
lda <- LDA(dtmTopicModeling, k = k, control = list(seed = seedNum))  # might take a few minutes...
```

We can now view the 'top 20' terms (words) for each of our 10 topics. The terms are ordered from most to least probable (how likely a term belongs to a particular topic), therefore the top terms are the most important for our analysis.

```r
# use the `terms` function to generate a dataframe of term
# probabilities from our LDA model
topTwentyTermsEachTopic <- terms(lda, 20)

# we can view this nicely using the `View` function
# View(topTwentyTermsEachTopic)

# you can also write the top 20 terms to CSV file in your R
# working directory:
write.csv(topTwentyTermsEachTopic, "top_20_terms_my_topic_model.csv",
    row.names = F)
```

Recall in the data cleaning process that we removed some rows which summed to zero (contained no terms after the cleaning process finished). When we want to analyse our comments dataset we need to remove these from our Vcorpus object and create a new Vcorpus object:

```r
# !! create a corpus of documents that exclude the documents
# we removed previously
facebookDataCorpus_LDA <- facebookDataCorpus[-toRemove]
```

How can we make sense of Topic 1?

We can generate a small random sample of 5 comments that belong to Topic 1 (i.e., this is the most likely topic for these comments). Then we can read and interpret the comments to help us make sense of why they were assigned to Topic 1. More importantly, we can start to figure out a label to describe Topic 1, and also come up with a brief summary of what that topic is.

```r
topicsProb <- topics(lda, 1)
topic1comments <- which(topicsProb == 1)
topic1commentsText <- as.list(facebookDataCorpus_LDA[topic1comments])
```

```r
set.seed(42)  # we set the seed number so we can replicate the random sample (if needed)
samplecomments <- sample(topic1commentsText, 5)
```

Here are the top 20 terms for Topic 1:

```r
topTwentyTermsEachTopic[, 1]
```

```
##  [1] "hair"        "prayers"     "rather"      "waited"      "kick"
##  [6] "open"        "ass"         "attitude"    "control"     "head"
## [11] "personal"    "examination" "survived"    "mind"        "agree"
## [16] "matter"      "choice"      "kit"         "little"      "colon"
```

Here are 5 sample comments that belong to Topic 1:

```r
lapply(samplecomments, function(x) {
    x[1]$content
})
```

```
## $`2935`
## [1] "I think it's great that they can give attention to people who are in the spotlight, I think it l
##
## $`3008`
## [1] "i pray she kicks its ass. i had surgery yesterday to remove cancer from my neck. cancer is very
##
## $`929`
## [1] "It has been 6 years ago for me, still very fresh in my mind on a daily bases. I too shaved my he
##
## $`2681`
## [1] "Just went through what you are going through now (Sept 2014 to present). Had the surgeries, the
##
## $`2046`
## [1] "For the entire month of October my company is offering this Breast Self Examination kit as a Buy
```

We can do the same process, this time for Topic 7.

```r
topic7comments <- which(topicsProb == 7)
topic7commentsText <- as.list(facebookDataCorpus_LDA[topic7comments])
set.seed(1)
samplecomments <- sample(topic7commentsText, 5)
```

Top 20 terms for Topic 7:

```r
topTwentyTermsEachTopic[, 7]
```

```
##  [1] "pray"     "prayers"  "beat"     "sorry"    "horrible" "shannon"
##  [7] "praying"  "recovery" "strength" "hear"     "thru"     "leave"
## [13] "doherty"  "thinking" "anyone"   "lots"     "mine"     "luck"
## [19] "hang"     "send"
```

Extract five sample comments that belong to Topic 7:

```r
lapply(samplecomments, function(x) {
    x[1]$content
})
```

```
## $`1038`
## [1] "Praying for you Shannen.  I am a cancer survivor, and I agree, the unknown is the scariest part
##
## $`1197`
## [1] "Lots of Prayers for your Shannen...   Hang in there...  Stay Positive and Fight this Horrible Ca
##
## $`2054`
## [1] "I really think March should be breast cancer awareness month. Why make us stress and worry righ
##
## $`2861`
## [1] "I am so sorry, sweetheart. I'm a breast cancer survivor. Be fearless and do what you have to do
##
## $`951`
## [1] "I dont know what your going through but family support laughter and plenty of inner strength is
```

Following this process we are able to go back and forth between the topics, top terms within each topic, and the text comments in order to make sense of the results. We can follow this process in order to provide a qualitative (and of course quantitative) topic model analysis of any kind of text data.

Importantly, some topics may not appear to make sense, no matter how hard you try. Sometimes the topic model will generate what are commonly called 'junk topics'. It is normal to have a small percentage (say 5% to 10% of topics) that are junk topics. If you are getting too many junk topics, it might mean that you need to try out different values of `k` for the number of topics, and try out different approaches to cleaning and preparing the text data.

Conclusion

I hope you have found this tutorial helpful and beneficial for your work. Please contact me with any questions (Timothy.Graham@anu.edu.au).

Tim