UCLA Anderson
SCHOOL *of* MANAGEMENT

MGMTMFE 431:

Data Analytics and Machine Learning

Topic 7b: Clustering and Unsupervised Learning: Part II

Spring 2019

Professor Lars A. Lochstoer

Text Topic Modelling

Text Topic Modelling is an unsupervised learning algorithm that tries to extract the main topics from a set of underlying documents (the Corpus)

Suppose you want to classify documents according to the relevance of what you want to analyze

- For instance, you are interested in newspaper articles related to investments and growth of business
- But, you don’t want to pre-specify words that define such documents either (a) because you are not sure exactly what words that would be and/or (b) because you don’t want to bias the analysis with your prior views

LDA helps you extract the text topics that best ‘fits’ your corpus

- With the topics in hand, you can then perform classification of each document

Latent Dirichlet Allocation (LDA): Intuition

- Suppose you have the following set of sentences:
  - I like to eat broccoli and bananas.
  - I ate a banana and spinach smoothie for breakfast.
  - Chinchillas and kittens are cute.
  - My sister adopted a kitten yesterday.
  - Look at this cute hamster munching on a piece of broccoli.

- What is latent Dirichlet allocation? It's a way of automatically discovering ***topics*** that these sentences contain. For example, given these sentences and asked for 2 topics, LDA might produce something like

  - **Sentences 1 and 2**: 100% Topic A
  - **Sentences 3 and 4**: 100% Topic B
  - **Sentence 5**: 60% Topic A, 40% Topic B
  - **Topic A**: 30% broccoli, 15% bananas, 10% breakfast, 10% munching, … (at which point, you could interpret topic A to be about food)
  - **Topic B**: 20% chinchillas, 20% kittens, 20% cute, 15% hamster, … (at which point, you could interpret topic B to be about cute animals)

The question, of course, is: how does LDA perform this discovery?

The LDA Model

- In more detail, LDA represents documents as ***mixtures of topics*** that spit out words with certain probabilities. It assumes that documents are produced in the following fashion: when writing each document, you:

1. Decide on the number of words N the document will have (say, according to a Poisson distribution).

2. Choose a topic mixture for the document (according to a Dirichlet distribution over a fixed set of K topics). For example, assuming that we have the two food and cute animal topics above, you might choose the document to consist of 1/3 food and 2/3 cute animals.

3. Generate each word *w_i* in the document by:
   i. First picking a topic (according to the multinomial distribution that you sampled above; for example, you might pick the food topic with 1/3 probability and the cute animals topic with 2/3 probability).
   ii. Using the topic to generate the word itself (according to the topic's multinomial distribution). For example, if we selected the food topic, we might generate the word "broccoli" with 30% probability, "bananas" with 15% probability, and so on.

4. Assuming this generative model for a collection of documents, LDA then tries to backtrack from the documents to find a set of topics that are likely to have generated the collection.

The LDA Model: Example

Consider the following example:

- According to the above process, when generating some particular document D, you might
  1. Pick 5 to be the number of words in D.
  2. Decide that D will be 1/2 about food and 1/2 about cute animals.
  3. Pick the first word to come from the food topic, which then gives you the word “broccoli”.
  4. Pick the second word to come from the cute animals topic, which gives you “panda”.
  5. Pick the third word to come from the cute animals topic, giving you “adorable”.
  6. Pick the fourth word to come from the food topic, giving you “cherries”.
  7. Pick the fifth word to come from the food topic, giving you “eating”.
- So the document generated under the LDA model will be “broccoli panda adorable cherries eating”

UCLA Anderson SCHOOL of MANAGEMENT

The LDA Model

- In the LDA model, we use Bayesian methods to estimate
- The priors over topics and words within each topic obey the Dirichlet distribution:
  - We use the symmetric version of this distribution (α the same across *x*'s)
  - Think of the below *x*'s (words, topics) as probabilities. The *x*'s are between zero and one and sum to one..

$$f(x_1, \dots, x_K; \alpha) = \frac{1}{B(\alpha)} \prod_{i=1}^{K} x_i^{\alpha-1}, \quad B(\alpha) = \frac{\prod_{i=1}^{K} \Gamma(\alpha)}{\Gamma\left(\prod_{i=1}^{K} \alpha\right)}$$

- The prior distribution over topics is governed by α. The higher α is, the more likely each document is to contain a mixture of most of the topics instead of any single topic.
- The prior distribution of words within each topic is also governed by a Dirichlet distribution as above, but to distinguish we here instead denote alpha as η
  - A higher value of η denotes that each topic is likely to contain a mixture of most of the words and not any word specifically.
- Finally, we update beliefs based on Bayes' rule:

$$Posterior = \frac{likelihood \times prior}{marginal\ likelihood}$$

The LDA Model: Learning

- So now suppose you have a set of documents. You've chosen some fixed number of K topics to discover, and want to use LDA to learn the topic representation of each document and the words associated to each topic. How do you do this? One way (known as collapsed Gibbs sampling) is the following:

- Go through each document, randomly assign each word in the document to one of the K topics.

- Notice that this random assignment already gives you both topic representations of all the documents and word distributions of all the topics (albeit not very good ones).

- So to improve on them, for each document *d*…
  - Go through each word *w* in *d*…
    - And for each topic t, compute two things: 1) p(topic t | document d) = the proportion of words in document d that are currently assigned to topic t, and 2) p(word w | topic t) = the proportion of assignments to topic t over all documents that come from this word w. Reassign *w* a new topic, where we choose topic t with probability p(topic t | document d) * p(word w | topic t) (according to our generative model, this is essentially the probability that topic t generated word w, so it makes sense that we resample the current word's topic with this probability). (Also, I'm glossing over a couple of things here, in particular the use of priors in these probabilities.)
    - In other words, in this step, we're assuming that all topic assignments except for the current word in question are correct, and then updating the assignment of the current word using our model of how documents are generated.

- After repeating the previous step a large number of times, you'll eventually reach a rough steady state where your assignments are pretty good. So use these assignments to estimate the topic mixtures of each document (by counting the proportion of words assigned to each topic within that document) and the words associated to each topic (by counting the proportion of words assigned to each topic overall).

The LDA Model: Worked example

- The full background maths of the procedure is in “LDA White Paper”, posted on CCLE
- Let’s consider the DJIA Headline News files
    - Text database of headlines from the Dow-Jones news wire
- Let’s find the main topics of these news headlines using LDA
- We will use the R package “lda” for this

UCLAAnderson
SCHOOL of MANAGEMENT

Read and clean the data

- Note, in this case we are interested in individual headlines (as stories within a day are typically meaningfully different in terms of topics).
  - Thus we do not aggregate by day as one may want to do if estimating, say, daily sentiment.

```
➢ # Load data
➢ data <- read.csv('DJIA_Headline_News.csv', stringsAsFactors = FALSE)

➢ # just learn topics of top 2 headlines per day (to save time)
➢ data <- rbind(data$Top1, data$Top2)

➢ # Get rid of those pesky b's and backslashes you see if you inspect the raw data
➢ data <- gsub('b"|b\'|\\\\|\\"', "", data)

➢ # Get rid of all punctuation except headline separators, alternative to cleaning done in tm-package
➢ data <- gsub("([<>])|[[:punct:]]", "\\1", data)
```

UCLAAnderson
SCHOOL of MANAGEMENT

LDA needs specific input format

```
# lda routine requires documents to be in specific list format:
# A list whose length is equal to the number of documents, D. Each element of
# documents is an integer matrix with two rows. Each column of
# documents[[i]] (i.e., document i) represents a word occurring in the document.
# documents[[i]][1, j] is a 0-indexed word identifier for the jth word in document
# i. That is, this should be an index - 1 into vocab. documents[[i]][2, j] is
# an integer specifying the number of times that word appears in the document.

# create large list for lda routine, split (tokenize) on space
doc.list <- strsplit(data, "[[:space:]]+")

# make all letters lowercase
doc.list <- sapply(doc.list, tolower)

# create a table of terms
term.table <- table(unlist(doc.list))
term.table <- sort(term.table, decreasing = TRUE)
head(term.table)

➢  the  to   of   in    a   and
➢ 3030 2448 1991 1874 1568 1269

# remove terms that are stop_words or occur less than 2 times (I just wanted you to see this option)
del <- names(term.table) %in% stopwords(kind = 'en') | term.table < 2
term.table <- term.table[!del]
head(term.table)

➢  us  will  new  says  police  people
➢ 354  242   192  190    186     185
```

UCLA Anderson SCHOOL of MANAGEMENT

LDA needs specific input format (cont'd)

```
# list of words in corpus, for use by lda
vocab = names(term.table)

# define function that helps getting the text data in right format for lda
get.terms <- function(x) {
            + index <- match(x, vocab) + index <- index[!is.na(index)]
             + rbind(as.integer(index - 1), as.integer(rep(1,length(index))))  }

# create list version of data
documents <- lapply(doc.list, get.terms)

# Compute some statistics related to the data set
D <- length(documents) # number of documents (3978)
W <- length(vocab) # number of terms in the vocab (5608)
doc.length <- sapply(documents, function(x) sum(x[2, ])) # number of tokens per document [11,3, 4...]
head(doc.length,20)
 [1] 8 1 11 5 5 4 8 9 4 7 8 7 9 4 12 7 14 17 14 8

N <- sum(doc.length) # total number of tokens/terms in the data

term.frequency <- as.integer(term.table) # Frequency of each of the terms
term.frequency

➢ [1] 354 242 192 190 186 185 184 169 142 129 128 123 122 118 113 111 107 100 96 95 94 91 90 90 89 89 89 88 88
➢ [30] 88  87  85  85  84  84  84  83  82  80  80  79  79  77  76  74  73  73 73 73 73 71 70 69 68 67 67 66 65
➢ , etc
```

UCLAAnderson
SCHOOL of MANAGEMENT

Fit the LDA model

```
# lda model tuning parameters
K <- 5 # number of topics
G <- 5000 # Number of iterations to arrive at convergence
alpha <- 0.02 # prior parameter for topics
eta <- 0.02 # prior parameter for words within each topic

#
# Fit the model
#
set.seed(357)
# Capture start time from your system
t1 <- Sys.time()

# Begin lda execution
lda_fit <- lda.collapsed.gibbs.sampler(documents = documents, K = K, vocab = vocab,
            + num.iterations = G, alpha = alpha, eta = eta, initial = NULL, burnin = 0,
            + compute.log.likelihood = TRUE)

# Capture end-time from your system
t2 <- Sys.time()

# Notice the time it took to complete lda on this data
# (you can use this technique whenever you want to check the execution time for an algorithm/piece of code)
t2 - t1 # outputs time it took to fit model
```

➢ Time difference of 58.14781 secs

UCLAAnderson
SCHOOL of MANAGEMENT

Key LDA model outputs

document_sums

- A K × D matrix where each entry is an integer indicating the number of times words in each document (column) were assigned to each topic (column).

topics

- A K × V matrix where each entry indicates the number of times a word (column) was assigned to a topic (row). The column names should correspond to the vocabulary words given in *vocab.*

UCLA Anderson
SCHOOL of MANAGEMENT

View topics

```
top.words <- top.topic.words(lda_fit$topics, num.words = 10, by.score = FALSE)
top.words
```

```
➢            [,1]       [,2]         [,3]        [,4]        [,5]
➢ [1,]      "new" "police"         "us"      "drug"        "us"
➢ [2,]    "years"    "man"        "law“        "us"    "israel"
➢ [3,]      "one"  "women" "government"      "says"   "israeli"
➢ [4,]  "million"  "woman"   "internet"     "drugs"     "north"
➢ [5,]     "will"  "years"       “will"     "world"     "korea"
➢ [6,]   "people"  "saudi"        "new"       "war“       "war"
➢ [7,]    "found"  "death"   "wikileaks"    "money"    "russia"
➢ [8,]    "first"   "girl"         "uk" "marijuana" "military"
➢ [9,]    "water"    "two"    "snowden"      "will"      "news"
➢ [10,]     "oil"  "court"       "says"       "tax"   "russian"
```

```
# show top documents per topic
top.documents <- top.topic.documents(lda_fit$document_sums, num.documents = 10, alpha = alpha)
top.documents
```

```
➢        [,1] [,2] [,3] [,4] [,5]
➢ [1,]   3335  696 2635 2727 3494
➢ [2,]   3357  486 3728 2155  570
➢ [3,]   3718  355 1745 3508  815
➢ [4,]   2854  894 1021 1670  924
➢ [5,]    465 2205 1598 2284 2069
➢ [6,]   2767 2291 2542 2951 1750
➢ [7,]   2080  288 2546  824 1293
➢ [8,]   2448  584 2781 2077   34
➢ [9,]   3095 1873 3094 1835 2430
➢ [10,] 3235 2116 3195 2010  231
```

UCLAAnderson
SCHOOL of MANAGEMENT

View most representative documents per topic

```
➢ # look at top documents in each topic (most representative)
data[top.documents[1,1]]
 [1] "There was no tape draped across a finish line but NASA is celebrating a win The agencys Mars Exploration
Rover Opportunity completed its first Red Planet marathon Tuesday 26219 miles 42195 kilometers with a finish
time of roughly 11 years and two months“


data[top.documents[1,2]]
[1] "Incredible Article On The Man Who Shot The Romanian Dictator Ceasescu And His Wife 20 Years Ago The
verdict was read out after a few hours The Ceausescus were sentenced to death They had ten days to appeal
but the sentence was to be carried out immediately A nod to Kafka“


data[top.documents[1,3]]
[1] "NSA infiltrates links to Yahoo Google data centers worldwide Snowden documents say The NSA has secretly
broken into the main communications links that connect Yahoo and Google data centers around the world according
to documents obtained from former NSA contractor Snowden”


data[top.documents[1,4]]
[1] "Golden age of antibiotics set to end We cannot say we werent warned The growing threat of antibiotic
resistant organisms is once again in the spotlight Prof Jeremy Farrar the new head of Britains biggest
medical research charity The Wellcome Trust said it was a truly global issue“


data[top.documents[1,5]]
[1] "Retired General Drones Create More Terrorists Than They Kill Iraq War Helped Create ISIS Retired Army
Gen Mike Flynn a top intelligence official in the post911 wars in Iraq and Afghanistan says in a forthcoming
interview on Al Jazeera English that drones do more harm than good"
```

UCLAAnderson
SCHOOL of MANAGEMENT

Get similarity of documents

- The topic model suggests a simple measure of similarity between two documents in terms of their themes
  - First get the predicted probability that, from the list of words in *vocab,* a word will show up in each document
  - This is based on the topic classification of that document
  - Think of this as the predictive distribution of the next word in that document if the author had written one more word

```
# get the predictive model probability that a word will show up in document
pred.dist <- predictive.distribution(lda_fit$document_sums, lda_fit$topics, alpha = alpha, eta = eta)
# words in rows, document in columns
pred.dist[1:10,1:8]
                        [,1]         [,2]         [,3]         [,4]         [,5]         [,6]         [,7]         [,8]
      us         0.0140483649 0.0153770555 1.606962e-02 0.0150867925 0.0024251415 1.593955e-02 0.0149166324 0.0155522903
      will       0.0045485980 0.0048560325 4.815458e-03 0.0053987026 0.0036928540 4.823078e-03 0.0055449550 0.0051403696
      new        0.0015382416 0.0019449589 1.742009e-03 0.0029340812 0.0015870305 1.780124e-03 0.0032183320 0.0024002805
      says       0.0045522186 0.0048567867 4.918680e-03 0.0050447434 0.0027326565 4.907056e-03 0.0050849912 0.0049923234
      police     0.0019481582 0.0003636932 3.810461e-05 0.0007765000 0.0128266882 9.925173e-05 0.0009281417 0.0004361878
      people     0.0052914904 0.0053821780 5.486480e-03 0.0048167890 0.0036050296 5.466891e-03 0.0046559135 0.0051162017
      government 0.0034147508 0.0038972546 3.895270e-03 0.0047813905 0.0017281699 3.895642e-03 0.0050108830 0.0043917908
      world      0.0047990856 0.0053478919 5.483082e-03 0.0050109669 0.0007005685 5.457693e-03 0.0049050900 0.0052250267
      years      0.0008694753 0.0003022405 3.201470e-05 0.0001402019 0.0053905989 8.276442e-05 0.0001352228 0.0000795812
      israel     0.0126317772 0.0134893859 1.443850e-02 0.0114644879 0.0002321613 1.426025e-02 0.0108094511 0.0128176353

# note that this can easily be used as a similarity measure to gauge similarity of two documents, e.g.:
sim_measure12 <- t(pred.dist[,1]) %*% pred.dist[,2]
sim_measure12
0.001820982
sim_measure13 <- t(pred.dist[,1]) %*% pred.dist[,3]
sim_measure13
0.001912684
```

Document 1 and 3 are more similar than document 1 and 2

UCLAAnderson
SCHOOL of MANAGEMENT

Probability that document is in a topic

- Finally, it may be useful to know the probability that a document is in any topic:

```
# get probability that each document belongs to a certain topic
## Number of documents to display
N <- 5
topic.proportions <- t(lda_fit$document_sums) / colSums(lda_fit$document_sums)
topic.proportions <- topic.proportions[sample(1:dim(topic.proportions)[1], N),]
topic.proportions[is.na(topic.proportions)] <- 1 / K
colnames(topic.proportions) <- apply(top.words, 2, paste, collapse=" ")
topic.proportions.df <- melt(cbind(data.frame(topic.proportions),document=factor(1:N)),
+variable.name="topic",id.vars = "document")
```

```
> topic.proportions.df
   document                                                     topic      value
1         1       new.years.one.million.will.people.found.first.water.oil 0.00000000
2         2       new.years.one.million.will.people.found.first.water.oil 0.16666667
3         3       new.years.one.million.will.people.found.first.water.oil 0.00000000
4         4       new.years.one.million.will.people.found.first.water.oil 0.00000000
5         5       new.years.one.million.will.people.found.first.water.oil 0.00000000
6         1       police.man.women.woman.years.saudi.death.girl.two.court 0.00000000
7         2       police.man.women.woman.years.saudi.death.girl.two.court 0.11111111
8         3       police.man.women.woman.years.saudi.death.girl.two.court 1.00000000
9         4       police.man.women.woman.years.saudi.death.girl.two.court 0.55555556
10        5       police.man.women.woman.years.saudi.death.girl.two.court 0.00000000
11        1  us.law.government.internet.will.new.wikileaks.uk.snowden.says 0.00000000
12        2  us.law.government.internet.will.new.wikileaks.uk.snowden.says 0.38888889
13        3  us.law.government.internet.will.new.wikileaks.uk.snowden.says 0.00000000
14        4  us.law.government.internet.will.new.wikileaks.uk.snowden.says 0.00000000
15        5  us.law.government.internet.will.new.wikileaks.uk.snowden.says 0.90909091
16        1         drug.us.says.drugs.world.war.money.marijuana.will.tax 0.00000000
17        2         drug.us.says.drugs.world.war.money.marijuana.will.tax 0.33333333
18        3         drug.us.says.drugs.world.war.money.marijuana.will.tax 0.00000000
19        4         drug.us.says.drugs.world.war.money.marijuana.will.tax 0.44444444
20        5         drug.us.says.drugs.world.war.money.marijuana.will.tax 0.09090909
21        1 us.israel.israeli.north.korea.war.russia.military.news.russian 1.00000000
22        2 us.israel.israeli.north.korea.war.russia.military.news.russian 0.00000000
23        3 us.israel.israeli.north.korea.war.russia.military.news.russian 0.00000000
24        4 us.israel.israeli.north.korea.war.russia.military.news.russian 0.00000000
25        5 us.israel.israeli.north.korea.war.russia.military.news.russian 0.00000000
```

Epilogue

- It is often useful to classify data into a lower dimensional space
  - E.g., Principal Components Analysis
- This can help make sense of what otherwise may look like just noise
- Data-cleaning is always very important
- Also, you will need to work on the exact topic specifications to get useful results.
  - You have to ‘get your hands dirty’ with whatever data you are analyzing
- With persistence, building a carefully tuned model along these lines can be tremendously useful