Lab 3: Topic Analysis

Javier Patrón

# Assignment Lab 3:

#### Due in 2 weeks: May 2 at 11:59PM

For this assignment you’ll the articles data you downloaded from Nexis Uni in Week 2.

## 1. Create a corpus from your articles.

## 2. Clean the data as appropriate. (Clean the data with stopwords)

tokens <- tokens(corpus, remove\_punct = T, remove\_numbers = T)  
add\_stops <- c(stopwords("en"))  
tokens\_selected <- tokens\_select(tokens, pattern = add\_stops, selection = "remove")

## 3. Run three models (i.e. with 3 values of k) and select the overall best value for k (the number of topics) - include some justification for your selection: theory, FindTopicsNumber() optimization metrics, interpretability, LDAvis. Select the best single value of k. Create optimization metrics (Understand the topics), and create 3 values of key.

### Steps:

#### a) Create a document-feature matrix (DFM) from tokenized text data.

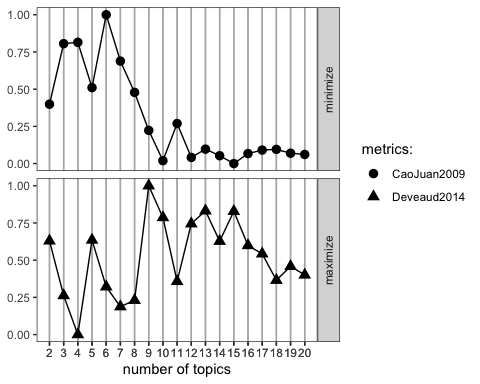
Each row will represent a document, and each column will represent a word or term. The values in the matrix ill represent the frequency of each feature in each document #### b) Trim the matrix to remove infrequently occurring terms. This matrix will be trimmed containing words or terms that ocurr more than min\_docfreq=x amount #### c) The resulting matrix is then subset to include only documents that contain at least that X term. #### d) Set the number of topics (k). One way to determine this is to identify the main themes or ideas in the text data and set k equal to the number of themes. #### e) Apply the Latent Dirichlet Allocation (LDA). This function LDA() creates an algorithm to the document-feature matrix (DFM) to extract the specified number of topics (k). The “Gibbs” method is used to estimate the model. The control argument specifies the number of iterations to run the algorithm for. The verbose argument is set to 25 to print out the status of the model estimation every 25 iterations.

# a) Create the document-feature matrix with the texts  
matrix <- dfm(tokens\_selected, tolower = T)   
  
# b) Trim the DFM to remove infrequently occurring terms  
matrix\_trimmed <- dfm\_trim(matrix, min\_docfreq = 30)  
  
# c) Subset the DFM to keep only documents with at least one term  
sel\_idx <- slam::row\_sums(matrix\_trimmed) > 0   
matrix\_trimmed <- matrix\_trimmed[sel\_idx,]  
  
# d) Set the number of topics  
topic\_number <- FindTopicsNumber(matrix\_trimmed,   
 topics = seq(from = 2,  
 to = 20,   
 by = 1),   
 metrics = c("CaoJuan2009", "Deveaud2014"),  
 method = "Gibbs",  
 verbose = T )

## fit models... done.  
## calculate metrics:  
## CaoJuan2009... done.  
## Deveaud2014... done.

FindTopicsNumber\_plot(topic\_number)

## Warning: The `<scale>` argument of `guides()` cannot be `FALSE`. Use "none" instead as  
## of ggplot2 3.3.4.  
## ℹ The deprecated feature was likely used in the ldatuning package.  
## Please report the issue at <]8;;https://github.com/nikita-moor/ldatuning/issueshttps://github.com/nikita-moor/ldatuning/issues]8;;>.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.



# Seeing the graph 8 topics seems like the correct K number   
k <- 5  
  
# e) Apply the Latent Dirichlet Allocation algorithm to the trimmed document-feature matrix. The resulting model will have k topics, specified by the variable k.  
  
topicModel\_k5 <- LDA(matrix\_trimmed, k,   
 method = "Gibbs", # The "Gibbs" method is used to estimate the model  
 control = list(iter = 500), # The control argument specifies the number of iterations to run the algorithm for  
 verbose = 25) # The verbose argument prints out the status of the model estimation every 25 iterations

## 4. Plot the top terms in each topic and the distribution of topics across a sample of the documents (constrained by what looks good in the plot).

### Steps:

#### a) Examine at our results.

#### b) Tidy the results

#### c) Extract the top words releated to the topics

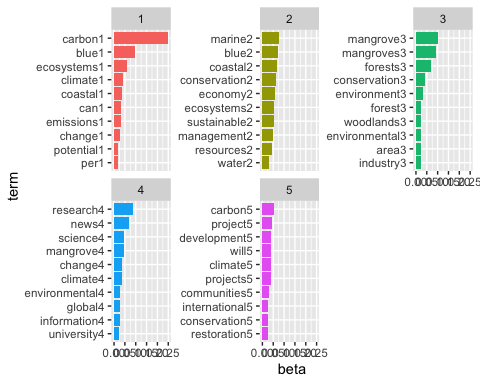
#### d) Visualize the results in a plot

#Calculating the posterior distribution of the topic model   
tmResult <- posterior(topicModel\_k5)  
  
# Prints the top 5 terms for each of the topics in the topicModel\_k5 model.  
terms(topicModel\_k5, 5)

## Topic 1 Topic 2 Topic 3 Topic 4 Topic 5   
## [1,] "carbon" "marine" "mangrove" "research" "carbon"   
## [2,] "blue" "blue" "mangroves" "news" "project"   
## [3,] "ecosystems" "coastal" "forests" "science" "development"  
## [4,] "climate" "conservation" "conservation" "mangrove" "will"   
## [5,] "coastal" "economy" "environment" "change" "climate"

# Extracting the matrix of document-topic probabilities from tmResult.  
# Theta shows how likely that document is to be associated with each of the topics in the model. The theta matrix can be used to identify the most relevant topics for a particular document.  
theta <- tmResult$topics  
  
# Extracting the matrix of topic-term probabilities from tmResult.  
# Beta shows how likely each word in the vocabulary is to be associated with that topic. Is the relationship between the words in the vocabulary and the topics in the model.  
beta <- tmResult$terms  
  
# Extracting the column names (i.e., the vocabulary) of the beta matrix and storing it in vocab.  
vocab <- (colnames(beta))

# b) Creating a "tidy" data frame from the topic-term matrix (also known as the "beta" matrix)   
main\_five\_topics <- tidy(topicModel\_k5, matrix = "beta")  
  
# c) Extracting the top 10 terms for each of the five topics in a topic model and arranging them in a tidy data frame.  
top\_blue\_carbon\_terms <- main\_five\_topics %>%  
 group\_by(topic) %>%  
 top\_n(10, beta) %>%  
 ungroup() %>%  
 arrange(topic, -beta)  
  
# d) Visualize the most significant terms for each topic in a blue carbon analysis  
top\_blue\_carbon\_terms %>%  
 mutate(term = reorder\_within(term, beta, topic, sep = "")) %>%  
 ggplot(aes(term, beta, fill = factor(topic))) +  
 geom\_col(show.legend = FALSE) +  
 facet\_wrap(~ topic, scales = "free\_y") +  
 scale\_x\_reordered()+  
 coord\_flip()



The highest beta values can be interpreted as the most salient words for a given topic, as they are the words that are most strongly associated with that topic in the corpus. Conversely, the lowest beta values can be interpreted as words that are not very relevant to the topic, as they are not strongly associated with it.

## 5. Take a stab at interpreting the resulting topics. What are the key themes discussed in the articles in your data base? Use the words, use what you know about the articles, and do an interpretation of what they mean.

### Steps:

#### a) Assign the names to the topics

#### b) Wrangle the data to plot

#### c) Plot the data and use the

To help us identify the topics we’re working with, we can assign names to them.

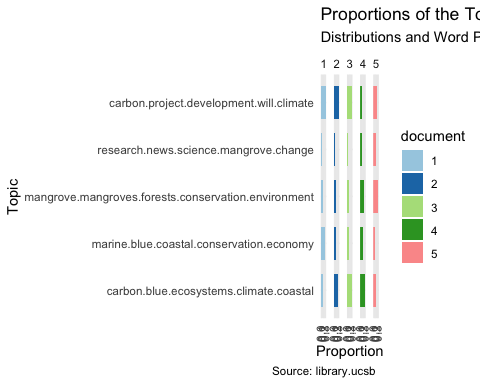
# a)  
topic\_words <- terms(topicModel\_k5, 5)  
topic\_names <- apply(topic\_words, 2, paste, collapse = " ")

We can explore the theta matrix, which contains the distribution of each topic over each document.

# b) Reshape the data in a format that can be visualized using ggplot  
  
# Create a vector of example document IDs from 1 to 5, and get the length of the vector  
example\_ids <- c(1:5)  
  
# Extract the topic proportions from the theta matrix for the example documents  
n <- length(example\_ids)  
  
# Assign the topic names as column names to the example\_props matrix  
example\_props <- theta[example\_ids,]  
  
# Reassign the names to the matrix  
colnames(example\_props) <- topic\_names  
  
  
# Use the melt function to convert the matrix to a long format, and create additional columns.   
viz\_df <- melt(cbind(data.frame(example\_props),  
 document = factor(1:n),  
 variable.name = "topic",  
 id.vars = "document"))

## Using document, variable.name, id.vars as id variables

# c) Create a bar chart with ggplot, using the reshaped data  
my\_palette <- brewer.pal(5, "Paired")  
  
ggplot(data = viz\_df, aes(x = variable,   
 y = value,   
 fill = document)) +  
 geom\_bar(stat="identity", width = 0.7) +  
 scale\_fill\_manual(values = my\_palette) +  
 labs(x = "Topic",   
 y = "Proportion",   
 title = "Proportions of the Top Topics",  
 subtitle = "Distributions and Word Proportions",  
 caption = "Source: library.ucsb ") +  
 theme\_minimal() +   
 theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1)) +  
 coord\_flip() +  
 facet\_wrap(~ document, ncol = n)



1. Here’s a neat JSON-based model visualizer. {LDAviz} is a helpful tool that uses JSON to visualize the distribution of words on topics and the distance between topics. The circles on the LDAvis plot are sized proportionally to the number of words that belong to each topic, and the distance between the circles represents how much they share words.

library(LDAvis) #visualization   
library("tsne") #matrix decomposition  
svd\_tsne <- function(x) tsne(svd(x)$u)  
json <- createJSON(  
 phi = tmResult$terms,   
 theta = tmResult$topics,   
 doc.length = rowSums(matrix\_trimmed),   
 vocab = colnames(matrix\_trimmed),   
 term.frequency = colSums(matrix\_trimmed),  
 mds.method = svd\_tsne,  
 plot.opts = list(xlab="", ylab="")  
)

## sigma summary: Min. : 33554432 |1st Qu. : 33554432 |Median : 33554432 |Mean : 33554432 |3rd Qu. : 33554432 |Max. : 33554432 |

## Epoch: Iteration #100 error is: 23.5745218796986

## Epoch: Iteration #200 error is: 0.110764783374565

## Epoch: Iteration #300 error is: 0.110550931230512

## Epoch: Iteration #400 error is: 0.110463281007594

## Epoch: Iteration #500 error is: 0.110463223111706

## Epoch: Iteration #600 error is: 0.110463222896666

## Epoch: Iteration #700 error is: 0.110463222534081

## Epoch: Iteration #800 error is: 0.110463222032521

## Epoch: Iteration #900 error is: 0.110463221395771

## Epoch: Iteration #1000 error is: 0.110463220625474

serVis(json)

## Loading required namespace: servr