IPSA NUS - Day 4

Topic Classification - Naïve Bayes Classifier

Dani Madrid-Morales (with code from content-analysis-with-r.com)

In today's class we will learn the process of using a pre-labeled dataset to train a Naïves Bayes Classifier. We will have a "simple" example, and then we'll work through a much more complex one that reproduces all the stages, decisions and nuances of doing machine learning.

The process of building a classifier works like this:
1. You will need to have some documents that are already classified. That is, documents with labels.
2. You will split those documents in 2 groups training set and testing set.
3. You will apply the nb classifier on the first group, and test it on the second group.
4. You will repeat the process as many times as needed until you are good to go.

Part 1 - Basic Naïve Bayes Classifier

We will need some new packages to run today's code, so let's get them all at once. You might need to uncomment the first few lines if you don't have those packages on your own computer. Make sure you have devtools in your computer before you proceed. Check the results carefully, as missing one single package will affect the overall performance of the code.

Let's start with a classifier of comments posted on reddit, a social media platform where users can post pretty much anything on almost any topic using subreddits. We are going to use data from two very subreddits, one about the Civil War in Syria, and one about Science. The data, which is available from https://content-analysis-with-r.com/, is saved in a file called `reddit.RData`

```
# The file contains a quanteda corpus, and the full dataset
load("reddit.RData")
glimpse(reddit.stats)
```

```
## Rows: 20,603
## Columns: 20
## $ Text             <fct> science1, science2, science3, science4, scien...
## $ Types            <int> 41, 23, 15, 33, 53, 3, 7, 66, 42, 38, 22, 5, ...
## $ Tokens           <int> 53, 30, 16, 44, 75, 3, 10, 84, 52, 57, 26, 5,...
## $ Sentences        <int> 7, 3, 2, 4, 6, 1, 3, 4, 2, 3, 2, 1, 1, 1, 6, ...
## $ structure        <chr> "1", "1_1", "1_1_1", "1_2", "1_2_1", "1_2_1_1...
## $ post_date        <date> 2009-12-01, 2009-12-01, 2009-12-01, 2009-12-...
## $ comm_date        <date> 2009-12-01, 2009-12-02, 2009-12-02, 2009-12-...
## $ num_comments     <dbl> 3646, 3646, 3646, 3646, 3646, 3646, 3646, 364...
## $ subreddit        <chr> "science", "science", "science", "science", "...
## $ upvote_prop      <dbl> 0.89, 0.89, 0.89, 0.89, 0.89, 0.89, 0.89, 0.8...
## $ post_score       <dbl> 857, 857, 857, 857, 857, 857, 857, 857, 857, ...
## $ author           <chr> "peteyH", "peteyH", "peteyH", "peteyH", "pete...
## $ user             <chr> "[deleted]", "ionstorm66", "ac3raven", "syste...
## $ comment_score    <dbl> 407, 192, 11, 51, 37, 9, 11, 6, 4, 3, 187, 98...
## $ controversiality <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ title            <chr> "What is the single scientific discovery you ...
## $ post_text        <chr> "Please identify one and elaborate if you hav...
## $ link             <chr> "https://www.reddit.com/r/science/comments/a9...
## $ domain           <chr> "self.science", "self.science", "self.science...
## $ URL              <chr> "http://www.reddit.com/r/science/comments/a9z...
```

```
reddit.corpus <- corpus(reddit.stats, text_field = "post_text")
reddit.corpus
```

```
## Corpus consisting of 20,603 documents and 19 docvars.
## text1 :
## "Please identify one and elaborate if you have to. Upvote tho..."
##
## text2 :
## "Please identify one and elaborate if you have to. Upvote tho..."
##
## text3 :
## "Please identify one and elaborate if you have to. Upvote tho..."
##
## text4 :
## "Please identify one and elaborate if you have to. Upvote tho..."
##
## text5 :
## "Please identify one and elaborate if you have to. Upvote tho..."
##
## text6 :
## "Please identify one and elaborate if you have to. Upvote tho..."
##
## [ reached max_ndoc ... 20,597 more documents ]
```

```
head(docvars(reddit.corpus), 5)
```

```
##       Text Types Tokens Sentences structure  post_date  comm_date
## 1 science1    41     53         7         1 2009-12-01 2009-12-01
## 2 science2    23     30         3       1_1 2009-12-01 2009-12-02
## 3 science3    15     16         2     1_1_1 2009-12-01 2009-12-02
## 4 science4    33     44         4       1_2 2009-12-01 2009-12-01
## 5 science5    53     75         6     1_2_1 2009-12-01 2009-12-01
##   num_comments subreddit upvote_prop post_score author        user
## 1         3646   science        0.89        857 peteyH   [deleted]
## 2         3646   science        0.89        857 peteyH  ionstorm66
## 3         3646   science        0.89        857 peteyH    ac3raven
## 4         3646   science        0.89        857 peteyH systemghost
## 5         3646   science        0.89        857 peteyH   [deleted]
##   comment_score controversiality
## 1           407                0
## 2           192                0
## 3            11                0
## 4            51                0
## 5            37                0
##                                                                          title
## 1 What is the single scientific discovery you most want to see in your lifetime?
## 2 What is the single scientific discovery you most want to see in your lifetime?
## 3 What is the single scientific discovery you most want to see in your lifetime?
## 4 What is the single scientific discovery you most want to see in your lifetime?
## 5 What is the single scientific discovery you most want to see in your lifetime?
##                                                                                                  link
## 1 https://www.reddit.com/r/science/comments/a9zdb/what_is_the_single_scientific_discovery_you_most/
## 2 https://www.reddit.com/r/science/comments/a9zdb/what_is_the_single_scientific_discovery_you_most/
## 3 https://www.reddit.com/r/science/comments/a9zdb/what_is_the_single_scientific_discovery_you_most/
```

```
## 4 https://www.reddit.com/r/science/comments/a9zdb/what_is_the_single_scientific_discovery_you_most/
## 5 https://www.reddit.com/r/science/comments/a9zdb/what_is_the_single_scientific_discovery_you_most/
##         domain
## 1 self.science
## 2 self.science
## 3 self.science
## 4 self.science
## 5 self.science
##
## 1 http://www.reddit.com/r/science/comments/a9zdb/what_is_the_single_scientific_discovery_you_most/?r
## 2 http://www.reddit.com/r/science/comments/a9zdb/what_is_the_single_scientific_discovery_you_most/?r
## 3 http://www.reddit.com/r/science/comments/a9zdb/what_is_the_single_scientific_discovery_you_most/?r
## 4 http://www.reddit.com/r/science/comments/a9zdb/what_is_the_single_scientific_discovery_you_most/?r
## 5 http://www.reddit.com/r/science/comments/a9zdb/what_is_the_single_scientific_discovery_you_most/?r
```

From the output above, we know that this corpus has over 20,000 documents. In the docvars, there's a column called `subreddit` that classifies the posts into either `sicence` or `syriancivilwar`. We are going to use this labels to train a model that would classify unseen comments on reddit into one of the two categories. Because this is a pretty large corpus, and it would take us a long time to train a model, we will work with a smaller subsample of 2,000 comments. The `corpus_sample` function in `quanteda` allows to do so.

```
# Random sample of 2,000 items in the Reddit corpus
set.seed(80) # Setting a seed guarantees next time I will get the same random sample
sample.reddit.corpus <- corpus_sample(reddit.corpus, size = 2000)
```

In this example we will split our labeled dataset into 1,500 comments as the training set that we will use to build a Naive Bayes classifier. In a second step, we will try to succesfully predict the sentiment for the remaining reviews (our test set).

```
# We generate 1500 numbers without replacement
id_train <- sample(1:2000, 1500, replace = FALSE) # Sample the range 1 to 2000, select 1500, do not rep
head(id_train, 10) # Show us the first 10 numbers you extracted
```

```
##  [1] 1694 1810   37 1300  927 1675  440 1992 1210 1621
```

The numbers above are the top 10 randomly extracted numbers. We are going to use these numbers to select documents in the corpus.

Next, we are going to create a new document variable (docvar) in my corpus. This variable ("id_numeric") is just a number that I can then compare to the numbers I randomly sampled. This way I could make the selection of cases that will be in my training dataset, and the cases that will be in my test dataset.

The outcome here should be two DFMs (remember, a DFM is a matrix with Documents and Features), one with the training set and one with the test set.

```
# Create docvar with ID (= number from 1 to 2000)
docvars(sample.reddit.corpus, "id_numeric") <- 1:ndoc(sample.reddit.corpus)

# Get training set by subsetting the corpus using randomly sampled numbers as criterion
dfmat_training <- corpus_subset(sample.reddit.corpus, id_numeric %in% id_train) %>%
  tokens(remove_symbols = TRUE, remove_punct = TRUE, remove_numbers = TRUE, remove_url = TRUE) %>%
  dfm(stem = TRUE) %>%
  dfm_remove(stopwords("english"))

# Get test set (documents not in id_train)
dfmat_test <- corpus_subset(sample.reddit.corpus, !id_numeric %in% id_train) %>%
  tokens(remove_symbols = TRUE, remove_punct = TRUE, remove_numbers = TRUE, remove_url = TRUE) %>%
  dfm(stem = TRUE) %>%
```

```
  dfm_remove(stopwords("english"))
```

Next we are ready to train the Naïve Bayes classifier using the function `textmodel_nb()`. This function takes two arguments: 1. The DFM of the training set 2. The variable (corpus metadata, or docvar in `quanteda`) that has the labels.

When we run this, the algorithm will compute the probability of word n being in category c. And, if we print the summary of the model, we will get a matrix, with words and categories.

```
tmod_nb <- textmodel_nb(dfmat_training, docvars(dfmat_training, "subreddit"))
summary(tmod_nb)
```

```
##
## Call:
## textmodel_nb.dfm(x = dfmat_training, y = docvars(dfmat_training,
##     "subreddit"))
##
## Class Priors:
## (showing first 2 elements)
##        science syriancivilwar
##            0.5            0.5
##
## Estimated Feature Scores:
##                   accord       lot     sourc      morn    offens    launch
## science        0.0001838 0.0001838 0.0001838 0.0001838 0.0001838 0.0001838
## syriancivilwar 0.0034671 0.0016836 0.0049507 0.0006001 0.0052674 0.0024670
##                      saa    n-hama    captur territori      held     rebel
## science        0.0001838 0.0001838 0.0001838 0.0001838 0.0001838 0.0001838
## syriancivilwar 0.0053841 0.0006001 0.0038672 0.0006001 0.0011835 0.0115682
##                    liltl       bit      free     time    moment   thought
## science        0.0001838 0.0001838 0.0001838 0.001471 0.0001838 0.0001838
## syriancivilwar 0.0006001 0.0011001 0.0047173 0.003634 0.0015669 0.0006001
##                     make megathread       map      peto     lucem
## science        0.0001838  0.0001838 0.0001838 0.0001838 0.0001838
## syriancivilwar 0.0022670  0.0019836 0.0067842 0.0011835 0.0011835
##                     area      hour       ago pro-regim     claim    latmin
## science        0.0001838 0.0001838 0.0001838 0.0001838 0.0001838 0.0001838
## syriancivilwar 0.0030504 0.0010668 0.0006001 0.0017669 0.0036338 0.0023503
##                    regim
## science        0.0001838
## syriancivilwar 0.0056174
```

Let's explore these results in a bit more detail. We are told that we ran a NB Classifier, assuming class priors of .5 and .5 (you could adjust that if needed with an additional argument in `textmodel_nb`). Then we get some words and, for each word, the probability assigned to it by the algorithm. These probabilites are the likelihood of a word appearing in documents of a certain label.

Naïve Bayes can only take features into consideration that occur both in the training set and the test set, but we can make the features identical by passing `training_dfm` to `dfm_match()` as a pattern.

```
dfmat_matched <- dfm_match(dfmat_test, features = featnames(dfmat_training))
```

We are now ready to use these probabilities to try to predict the label of the remaining documents in the dataset. So, remember, we divided our 2,000 labeled comments into a training and test dataset. Now we are going to ask `quanteda` to predict the categories of the test dataset ($n = 500$) and then we will compare these predictions to the actual labels that we have.

To apply an NB to an unseen dataset, we use the function `predict`. This takes two arguments:
1. Argument 1 is the result of the probabilites that we computed earlier (that is, our model). 2. A new dataset unseen by the algorithm that we want to classify.

```
# Create a simple dataframe with the TRUE labels of the test set that we can use later
actual_class <- docvars(dfmat_matched, "subreddit")

# Predict the categories using the NB model we trained
predicted_class <- predict(tmod_nb, newdata = dfmat_matched)
```

Let's inspect how well the classification worked. For this, we are creating a table with the actual labels we stored before, the results of the `predict` function.

```
tab_class <- table(actual_class, predicted_class)
tab_class
```

```
##                 predicted_class
## actual_class     science syriancivilwar
##   science            267              0
##   syriancivilwar      85            148
```

In the confusion matrix above, we see that the algorithm did relatively well. We can use the function confusionMatrix() from the `caret` package to assess the performance of the classification.

```
confusionMatrix(tab_class, mode = "everything")
```

```
## Confusion Matrix and Statistics
##
##                 predicted_class
## actual_class     science syriancivilwar
##   science            267              0
##   syriancivilwar      85            148
##
##                Accuracy : 0.83
##                  95% CI : (0.7941, 0.8619)
##     No Information Rate : 0.704
##     P-Value [Acc > NIR] : 5.714e-11
##
##                   Kappa : 0.6503
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7585
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.6352
##               Precision : 1.0000
##                  Recall : 0.7585
##                      F1 : 0.8627
##              Prevalence : 0.7040
##          Detection Rate : 0.5340
##    Detection Prevalence : 0.5340
##       Balanced Accuracy : 0.8793
##
##        'Positive' Class : science
##
```

The model seems to be relatively robust with measures of precision around .8. In a real research context, we would repeat this operation multiple times, we different folds of the training set and test set, until we reached an even higher level of accuracy, precision and recall. We would then use this one trained model to a larger unseen dataset.

Part 2 - Fitting a NB in the "real" world

Your job as a computational social scientist is unlikely to be this easy most of the time. Usually you will be faced with lots of choices and difficulties in getting all your data ready.

The next example is a *lot more complex* and requires more input on the side of the researcher to achieve a good score in the precision metrics.

There are a lot of required files for this exercise, so if you are going to run this on your own computer, make sure you have downloaded all of them, and that they are saved in the correct location. Otherwise you are likely to encounter errors as you go through the code.

The starting point here is to load the unlabeled set of documents. The file "cctvnewscontent_lab4.csv" includes the transcript of over 13,000 news stories broadcast on China's State Broadcaster CCTV's global branch, now called CGTN. We are going to try to classify these stories using an NB model trained using a smaller sample of pre-labeled data.

To get started, we read the document (a .csv file) using the `fread` function from the `data.table` package, which we have used before.

```
cctv.unlabeled <- fread("cctvnewscontent_lab4.csv")
```

The dataset contains over 230 variables that we do not need to explore right now. I'll bring your attention to the variable `wc` which has a wordcount for each transcript. Because really short texts (less than 50 words) are going to be very difficult to classify, we will remove those first.

After we subset the dataset, we are ready to call the `corpus` function in `quanteda`and pass the column name where the texts are stored as the source for the corpus.

```
# Uses base R to subset a dataframe, by year and wordcount (that is wc).
cctv_news <- subset(cctv.unlabeled, wc>50)
rm(cctv.unlabeled)

# Creates a simple quanteda corpus
cctv.corpus <- corpus(cctv_news$text)
```

We also want to create some metadata that might be useful for us at a later stage to do some analysis. In this next chunk of code, we are splitting up the dates into months, years, weeks and days. This can be done with the `lubridate` package.

The functions `year`, `month` and so on, take a full date (2019-12-03 o 2019/12/03 or 1-Dec-19, or pretty much any known format), and then extract the element that we are interested in.

We are then, adding this new variable as metadata of our newly created corpus.

```
#In the cctv.corpus, I add a new docvar called "date" which has the published date
docvars(cctv.corpus, "date") <- cctv_news$date.publish

#In the cctv.corpus, I add a new docvar called "year" which has the year extracted from the date
docvars(cctv.corpus, "year") <- year(docvars(cctv.corpus, 'date'))

# And so on.
docvars(cctv.corpus, "month") <- month(docvars(cctv.corpus, 'date'))
docvars(cctv.corpus, "week") <- week(docvars(cctv.corpus, 'date'))
docvars(cctv.corpus, "day") <- day(docvars(cctv.corpus, 'date'))
```

```
# We also add title as metadata in the corpus
docvars(cctv.corpus, "title") <- cctv_news$title

# And the locations cited in each news iem.
docvars(cctv.corpus, "locations") <- cctv_news$locations
```

We are now ready to tokenize our corpus. Remember, tokenizing separates each document into tokens (that is, words). We are also going to take this opportunity to clean the tokenized version of the corpus, by removing things like stopwords, punctuation and so on. To make the analysis easier for us, we are just

This might take a few seconds to run.

```
# We first tokenize our corpus and remove spaces, numbers and punctuation
cctv.toks <- tokens(cctv.corpus,
                    remove_separators = TRUE,
                    remove_numbers = TRUE,
                    remove_punct = TRUE)

# We remove stopwords, as well as some words that occurr too often such as CCTV and so on
cctv.toks <- tokens_remove(cctv.toks,
                           c(stopwords("english", "smart"),"CCTV")) %>%
  tokens_wordstem()
```

Let's have a look at one of the documents that has been tokenized, for example document 3433. We are going to print the first 25 words in this tokenized document.

```
head(cctv.toks[[3433]], 25)
```

```
##  [1] "Qinghai"     "Internat"    "Carpet"      "Exhibit"     "open"
##  [6] "Xine"        "capit"       "citi"        "northwest"   "China"
## [11] "Qinghai"     "Provinc"     "Monday"      "businessmen" "home"
## [16] "abroad"      "attend"      "annual"      "event"       "exhibit"
## [21] "area"        "booth"       "Tibetan"     "carpet"      "display"
```

As we know, the process of working with `quanteda` is that we first create a corpus, then we tokenize that corpus and then we create a DFM (document feature matrix) that can be used for most analyses. So, let's do that.

We are also going to lowercase our DFM, as we know this is an important step to reduce the dimensions of our DFM.

```
# Creates a DFM, with lowercased features (i.e. words)
cctv.dfm <- dfm(cctv.toks,
                tolower = TRUE)

# How many documents are there in my DFM?
ndoc(cctv.dfm)
```

```
## [1] 12940
```

```
# How many features (i.e. words) are there in my DFM?
nfeat(cctv.dfm)
```

```
## [1] 49120
```

So, there are 12,940 documents and more than 49.000 words. This is too large of a number, so we are going to trim this DFM to make more manageable. According to Zipf's law,words that do not occur too often aren't really "important" for QTA, so we can remove words that occur less than a certain number of times and that appear in less than a certain number of documents.

These values are somewhat arbitrary. We could see which are the top occurring words to get an idea of general frequencies, and use that to determine how much we are going to trim our DFM

```
topfeatures(cctv.dfm, 10)
```

```
##   china    year  chines   peopl countri provinc    citi    time   water
##   23846   13646   11988   10750    7941    7434    7165    6593    6528
##   local
##    6522
```

From this, we also see that China seems to be occuring very often. To correct for this, we could remove all mentions of countries from our DFM. Luckily for us, quanteda has a dictionary of countries in the package `newsmap` that we can pass to quanteda to remove the words.

```
# Removes mentions of a country
cctv.dfm <- dfm_remove(cctv.dfm,
                           dictionary(data_dictionary_newsmap_en))

# We can print the top features and see we let go of China and Chinese
topfeatures(cctv.dfm, 10)
```

```
##    year  chines   peopl countri provinc    citi    time   water   local
##   13646   11988   10750    7941    7434    7165    6593    6528    6522
##  govern
##    6233
```

```
# Trim the DFM to exclude words that occur less than 10 times
cctv.dfm <- dfm_trim(cctv.dfm, min_termfreq = 10)

# How many features (i.e. words) are there in my DFM after trimming?
nfeat(cctv.dfm)
```

```
## [1] 9274
```

We have gone from more than 49.000 featuresm to 9,274, which is enough to get diversity and specificity in our DFM.

Now that we finished getting our corpus of interest ready we need to train an algorithm that we can then apply to this corpus to see what topic each of the documents belongs to.

For this process, I need to create a dataframe that has 2 things:
1. The text of the documents that are labeled
2. The labels of these documents

To get to this point, we are going to collect some Word files that have the text, and combine them with an Excel file that has the labels.

Let's start with importing the Word files to R. To do this, I am going to make a list of all the files, then I am going to write a loop that load the file to R, reads it, extracts the text and pastes it into a single dataframe

```
# Create a list of file names
list.transcripts.paths <- list.files(path="Texts/")

# Creates an empty object where I am going to store my text files
full.transcripts2012 <- c()

# A for loop that goes through the list of files in and extracts the text
for(i in list.transcripts.paths){ # Go through the list of file names
  file <- paste("Texts/",i,collapse="", sep = "") # Create a path where the file is stored
  transcript <- read_docx(file) # Read the file to R
```

```
  transcript <- paste(transcript, collapse = " ") # Collapse or concatenate all the text into 1 long st
  full.transcripts2012 <- paste(transcript, full.transcripts2012, collapse = " ") # Add it to my empty
}
```

The object `list.transcripts.paths` is just a list of file names. The object `full.transcripts` is now a VERY long character with ALL the text from the 56 Word files. This is of no use to us, so we are going to split this LONG character into 1199 shorter texts, each of which has the text of 1 news story.

Luckily for us, the beginning and end of each document is clearly marked with the words "ID –" or "ID -". Therefore, next we are going to have R look for these instances of either phrase and add a marker ("|") that we are then going to use to split the character object into smaller chunks one for each news story.

```
# We look for instances of ID -- or ID - and use the marker | to substitute them
full.transcripts2012 <- str_replace_all(full.transcripts2012, fixed("ID -- "), "|")
full.transcripts2012 <- str_replace_all(full.transcripts2012, fixed("ID - "), "|")

# We tell R, split the long character "full.transcripts2012" into shorter strings using "|" to make the
a <- strsplit(full.transcripts2012, "[|]") # Creates a list

# We unlist "a" into a DF of one single column including each news item in one row
df <- data.frame(matrix(unlist(a), nrow=lengths(a), byrow=T))
colnames(df) <- "text" # Name this new column "text"
df$text <- str_replace_all(df$text, fixed("|"), "") # Remove the marker "|" from the text

# Each document has an ID number, and we are going extract it form the text
df$id <- substr(df$text,1,4) # Extract the first 4 characters of the text (which is where the ID is)
df$text <- substring(df$text,5) # Use the rest of content as the text

# Let's clean our R Environment and remove objects we no longer need
rm(a)
rm(full.transcripts2012)
rm(list.transcripts.paths)
rm(i)

# Let's have a look at what this new df looks like
head(df)
```

```
##
## 1
## 2  Peninah: Hello and welcome to Africa Live. I'm Peninah Karibe. Let's take you live to South Africa
## 3
## 4
## 5
## 6
##     id
## 1
## 2 1177
## 3 1178
## 4 1179
## 5 1180
## 6 1181
```

So we've created a `df` that has text and ID. Now we are missing the labels for each of these texts. These labels are stored in another Excel file that we are now going to merge to the one that includes the texts

```
# Read the CSV file with the labels
cgtn.dataset <- read.csv(file = "CCTVFullDatasetFromR.csv", colClasses=c("id"="character"))

# Remove the first column, which we don't need
cgtn.dataset$X <- NULL

# Merge the 2 datasets using "id" as the key to merge them
full.text.cgtn <- merge(cgtn.dataset[,c(1:106,407:414)], df[2:1200,], "id")

# Let's just keep the variables we are interested in
full.text.cgtn %<>%
  select(id,
           year,
           topic_name,
           text)

# Let's have a look at what the new full.text.cgtn df looks like
glimpse(full.text.cgtn)
```

```
## Rows: 1,199
## Columns: 4
## $ id         <chr> "0001", "0002", "0003", "0004", "0005", "0006", "00...
## $ year       <int> 2012, 2012, 2012, 2012, 2012, 2012, 2012, 2012, 201...
## $ topic_name <fct> "Military & Political Violence", "Crime, Legal & Ju...
## $ text       <chr> " We start in Syria where UN Arab League special en...
```

```
# Finally, let's clean our Environment of unnecessary variables
rm(cgtn.dataset)
rm(df)
rm(file)
rm(transcript)
```

So, our training dataset has 4 variables, one with the text ID, one with the year, one with the topic label assigned by human coders, and one with the text. Let's see which topics are included in this hand coded dataset

```
table(full.text.cgtn$topic_name)
```

```
##
## Crime, Legal & Judicial Matters     Culture, Religion & Tourism
##                              36                              41
##       Ecology & the Environment               Economic Matters
##                              28                             273
##       Foreign Affairs/Diplomacy       Human Interest & Oddities
##                             132                              20
##   Military & Political Violence   Natural Disasters & Accidents
##                             163                              22
##         Politics within a State            Science & Technology
##                             186                              15
##                  Social Issues                          Sports
##                              98                             185
```

From the table above, we see that there's some topics that do not occur very often, while other occur quite often. So, we might want to reduce the dimensions of our labels, to have just 3 groups:
1. Political news (include diplomacy, domestic politcs, and military and political violence, crime legal and judicial matters)

2. News about the economy (including economic matters, social issues and science and technology)
3. News about other topics

For this, we will need to recode our labels into a new variable, which we will code "label" that will take only 3 levels: politics, economy and others. To do this, we are going to use the `case_when` function, that allows us to give conditions to the recoding of a variable.

```
full.text.cgtn %<>%
  mutate(label = case_when(
    topic_name == "Politics within a State" ~ "politics",
    topic_name == "Military & Political Violence" ~ "politics",
    topic_name == "Foreign Affairs/Diplomacy" ~ "politics",
    topic_name == "Economic Matters"~ "economy",
    topic_name == "Science & Technology" ~ "economy",
    topic_name == "Social Issues" ~ "economy",
    TRUE ~ "sports & culture "))

table(full.text.cgtn$label)
```

```
##
##           economy          politics sports & culture
##               386               481               332
```

We now have three more or less balance categories, that we can use as labels to train our Naïve Bayes algorithm. Before we can do that, we need to turn this df with the texts and labels, into a corpus that quanteda is able to read for the NB classifier.

```
labeled.corpus <- corpus(full.text.cgtn)

# Let's add the labels we just created as docvars
docvars(labeled.corpus, "label") <- full.text.cgtn$label
```

As usual, let's tokenize the labeled dataset and do some cleaning by rmeoving punctuation, numbers, symbols and stop words, as well as the word CCTV which appears very often.

```
# We use "tokens" to tokenize the corpus
labeled.toks <- tokens(labeled.corpus,
                    remove_separators = TRUE,
                    remove_numbers = TRUE,
                    remove_punct = TRUE)

# We remove stopwords, as well as some words that occurr too often such as CCTV and so on
labeled.toks <- tokens_remove(labeled.toks,
                           c(stopwords("english", "smart"),"CCTV")) %>%
  tokens_wordstem()

# Let's have a look at the tokenized version of document 33
head(cctv.toks[[33]], 20)
```

```
##  [1] "Peopl"     "crystal"   "ball"      "foretel"   "futur"
##  [6] "convinc"   "project"   "call"      "Live"      "Earth"
## [11] "Simul"     "oper"      "intern"    "group"     "scientist"
## [16] "claim"     "predict"   "futur"     "accur"     "Live"
```

Notice how this process is always the same, and in this case, it is doing the exact same thing we did with the unlabeled dataset. We can now proceed to create a DFM, in which we will remove the names of countries (just as we did before), and we will explore the top occurring features.

```
# Creates a DFM, with lowercased features (i.e. words)
labeled.dfm <- dfm(labeled.toks,
                   tolower = TRUE)

# How many documents are there in my DFM?
ndoc(labeled.dfm)
```

```
## [1] 1199
```

```
# How many features (i.e. words) are there in my DFM?
nfeat(labeled.dfm)
```

```
## [1] 11202
```

In this labeled dataset, we currently have almost as many words as we did in our 20,000+ articles of the unlabeled dataset. We therefore might want to do the same process we did before of removing the names of countries (they are occurring too often, and also removing words that do not appear too often).

```
# Removes mentions of a country
labeled.dfm <- dfm_remove(labeled.dfm,
                          dictionary(data_dictionary_newsmap_en))

# We can print the top features and see how many there are in total
topfeatures(labeled.dfm, 10)
```

```
## countri  africa    year   peopl   south  govern african  presid    time
##    1424    1292    1094    1027     997     975     848     827     660
##   world
##     642
```

```
nfeat(labeled.dfm)
```

```
## [1] 10949
```

```
# Trim the DFM to exclude words that occur less than 10 times and see total number of features
labeled.dfm <- dfm_trim(labeled.dfm, min_termfreq = 5)
nfeat(labeled.dfm)
```

```
## [1] 3621
```

```
topfeatures(labeled.dfm, 10)
```

```
## countri  africa    year   peopl   south  govern african  presid    time
##    1424    1292    1094    1027     997     975     848     827     660
##   world
##     642
```

At this stage, we are ready to start the cross-validation of our model. We are going to run multiple iterations of the NB classifier until we reach a point that we think is good enough.

The process is just like we saw at the beginning of the lab. 1. We sample a % of labeled articles and we use them as training set. 2. We will then try to predict categories in the unseen dataset. 3. We will print the confusion matrix and repeat until we find a model that is a good fit.

```
# Iteration 1

# Divide the labeled dataset into training and test
id_train1 <- sample(1:1199, 800, replace = FALSE) # Random sampling

# Adding a variable named "id_numeric" to the DFM
```

```r
docvars(labeled.dfm, "id_numeric") <- 1:ndoc(labeled.dfm)

# Split the labeled dataset into training and test
training_dfm <- dfm_subset(labeled.dfm, id_numeric %in% id_train1)
test_dfm <- dfm_subset(labeled.dfm, !id_numeric %in% id_train1)

# Fit the NB classifier
nb <- textmodel_nb(training_dfm, docvars(training_dfm, "label"))
summary(nb, 50) # Prints probabilites of words per topic
```

```
##
## Call:
## textmodel_nb.dfm(x = training_dfm, y = docvars(training_dfm,
##     "label"))
##
## Class Priors:
## (showing first 3 elements)
##           economy          politics sports & culture
##            0.3333            0.3333            0.3333
##
## Estimated Feature Scores:
##                      start      arab     leagu   special     envoy
## economy           0.002386 6.449e-05 0.0001290 0.0004514 3.225e-05
## politics          0.001638 1.342e-03 0.0006143 0.0005688 2.730e-04
## sports & culture  0.002733 1.171e-04 0.0021083 0.0008980 3.904e-05
##                      crisi      kofi     annan     call    member
## economy           0.0010963 3.225e-05 3.225e-05 0.001322 0.0009351
## politics          0.0013651 1.138e-04 1.138e-04 0.002912 0.0015699
## sports & culture  0.0003123 3.904e-05 3.904e-05 0.001484 0.0008980
##                     secur   council   overcom   prevail  deadlock
## economy           0.001290 0.0001935 1.290e-04 3.225e-05 9.674e-05
## politics          0.006143 0.0013879 4.550e-05 6.826e-05 9.101e-05
## sports & culture  0.001171 0.0001171 7.809e-05 1.562e-04 3.904e-05
##                      adopt      unit     posit     enabl    mediat
## economy           0.0001935 0.0003547 0.0009996 2.257e-04 3.225e-05
## politics          0.0003640 0.0005005 0.0011149 4.550e-05 2.958e-04
## sports & culture  0.0001562 0.0003123 0.0013665 3.904e-05 3.904e-05
##                     effort     deliv      peac   address     press
## economy           0.0008706 0.0004192 0.0005159 0.0004837 9.674e-05
## politics          0.0012741 0.0002730 0.0036631 0.0006826 4.550e-04
## sports & culture  0.0006247 0.0001952 0.0005076 0.0001171 2.733e-04
##                     confer     meet      send   technic      team
## economy           0.0004192 0.001967 0.0002902 2.902e-04 0.0003869
## politics          0.0009556 0.002707 0.0008418 9.101e-05 0.0003868
## sports & culture  0.0005466 0.001054 0.0001562 4.685e-04 0.0081209
##                     prepar      talk    resolv      stay   announc
## economy           0.0002902 0.0019670 0.0001612 0.0006127 0.0009029
## politics          0.0006826 0.0037086 0.0007281 0.0007736 0.0013879
## sports & culture  0.0012103 0.0008589 0.0001562 0.0008980 0.0004685
##                        set      zone   protect      grow   number
## economy           0.0015478 2.902e-04 0.0007094 0.0015478 0.001999
## politics          0.0008418 2.048e-04 0.0008191 0.0007281 0.001479
## sports & culture  0.0016398 3.904e-05 0.0006247 0.0008589 0.001796
##                     refuge      flee   violent   presid    bashar
```

```
## economy           0.0006772 0.0001290 9.674e-05 0.002031 3.225e-05
## politics          0.0008418 0.0002958 2.275e-04 0.010193 9.101e-05
## sports & culture  0.0001171 0.0001171 3.904e-05 0.001523 3.904e-05
##                         al     assad      fled  northern      hour
## economy           9.674e-05 3.225e-05 9.674e-05 0.0001612 0.0007739
## politics          1.775e-03 4.550e-05 4.095e-04 0.0005461 0.0007736
## sports & culture  4.685e-04 3.904e-05 1.171e-04 0.0002343 0.0006247
```

The results above show us each word in our labeled dataset and the probability of this word belonging to one of 3 labels.

We would now be ready to try to predict the categories of the remaining 399 articles the algorithm hasn't seen. Before that, and as we learned earlier, we need to match the words appering in the training and test datasets because Naïve Bayes can only take features into consideration that occur both in the training set and the test set. We can fix this by passing training_dfm to dfm_match() as a pattern.

```
# Matches training and test sets
dfmat_matched <- dfm_match(test_dfm, features = featnames(training_dfm))

# Predicting categories in the unseen group of documents
pred_data <- predict(nb, test_dfm)

# Building a confusion matrix to assess the accuracy, precision and recall metrics
actual_class <- docvars(test_dfm, "label")
predicted_class <- pred_data
class_table <- table(actual_class, predicted_class)
confusionMatrix(class_table, mode = "everything")
```

```
## Confusion Matrix and Statistics
##
##                   predicted_class
## actual_class       economy politics sports & culture
##   economy              126        8                5
##   politics               9      138                2
##   sports & culture      14        8               89
##
## Overall Statistics
##
##                Accuracy : 0.8847
##                  95% CI : (0.8492, 0.9143)
##     No Information Rate : 0.386
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.825
##
##  Mcnemar's Test P-Value : 0.04765
##
## Statistics by Class:
##
##                      Class: economy Class: politics
## Sensitivity                  0.8456          0.8961
## Specificity                  0.9480          0.9551
## Pos Pred Value               0.9065          0.9262
## Neg Pred Value               0.9115          0.9360
## Precision                    0.9065          0.9262
## Recall                       0.8456          0.8961
```

```
## F1                           0.8750          0.9109
## Prevalence                   0.3734          0.3860
## Detection Rate               0.3158          0.3459
## Detection Prevalence         0.3484          0.3734
## Balanced Accuracy            0.8968          0.9256
##                      Class: sports & culture
## Sensitivity                           0.9271
## Specificity                           0.9274
## Pos Pred Value                        0.8018
## Neg Pred Value                        0.9757
## Precision                             0.8018
## Recall                                0.9271
## F1                                    0.8599
## Prevalence                            0.2406
## Detection Rate                        0.2231
## Detection Prevalence                  0.2782
## Balanced Accuracy                     0.9272
```

Our NB classifier seems to have done pretty well, with measures well above .8. This is a good starting point, but we want to repeat the process a few times to find the best model for our data. So, we will iterate the process from randomly splitting our dataset into 2 groups (training and test), to fitting the model and testing the precision metrics.

We will do this 4 more times. The code chunks below, repeat the code above and just change the seed, so we get a different sample of articles each time.

```
# Iteration 2

id_train2 <- sample(1:1199, 800, replace = FALSE)
docvars(labeled.dfm, "id_numeric") <- 1:ndoc(labeled.dfm)
training_dfm <- dfm_subset(labeled.dfm, id_numeric %in% id_train2)
test_dfm <- dfm_subset(labeled.dfm, !id_numeric %in% id_train2)
nb <- textmodel_nb(training_dfm, docvars(training_dfm, "label"))
summary(nb, 10)
```

```
##
## Call:
## textmodel_nb.dfm(x = training_dfm, y = docvars(training_dfm,
##     "label"))
##
## Class Priors:
## (showing first 3 elements)
##           economy          politics sports & culture
##            0.3333            0.3333            0.3333
##
## Estimated Feature Scores:
##                      start      arab     leagu   special     envoy
## economy           0.002427 0.0003782 0.0001576 0.0005358 3.152e-05
## politics          0.001577 0.0009461 0.0004731 0.0007884 2.253e-04
## sports & culture  0.002720 0.0001183 0.0020892 0.0008278 3.942e-05
##                      crisi      kofi     annan     call    member
## economy           0.0012293 3.152e-05 3.152e-05 0.001576 0.0008826
## politics          0.0012615 2.253e-05 2.253e-05 0.002703 0.0016220
## sports & culture  0.0002759 3.942e-05 3.942e-05 0.001222 0.0007490
```

```r
dfmat_matched <- dfm_match(test_dfm, features = featnames(training_dfm))
pred_data <- predict(nb, test_dfm)
actual_class <- docvars(test_dfm, "label")
predicted_class <- pred_data
class_table <- table(actual_class, predicted_class)
confusionMatrix(class_table, mode = "everything")
```

```
## Confusion Matrix and Statistics
##
##                    predicted_class
## actual_class        economy politics sports & culture
##   economy               123        7                 5
##   politics                6      128                 5
##   sports & culture       19        9                97
##
## Overall Statistics
##
##                Accuracy : 0.8722
##                  95% CI : (0.8354, 0.9033)
##     No Information Rate : 0.3709
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8077
##
##  Mcnemar's Test P-Value : 0.02457
##
## Statistics by Class:
##
##                      Class: economy Class: politics
## Sensitivity                  0.8311          0.8889
## Specificity                  0.9522          0.9569
## Pos Pred Value               0.9111          0.9209
## Neg Pred Value               0.9053          0.9385
## Precision                    0.9111          0.9209
## Recall                       0.8311          0.8889
## F1                           0.8693          0.9046
## Prevalence                   0.3709          0.3609
## Detection Rate               0.3083          0.3208
## Detection Prevalence         0.3383          0.3484
## Balanced Accuracy            0.8916          0.9229
##                      Class: sports & culture
## Sensitivity                           0.9065
## Specificity                           0.9041
## Pos Pred Value                        0.7760
## Neg Pred Value                        0.9635
## Precision                             0.7760
## Recall                                0.9065
## F1                                    0.8362
## Prevalence                            0.2682
## Detection Rate                        0.2431
## Detection Prevalence                  0.3133
## Balanced Accuracy                     0.9053
```

```r
# Iteration 3

id_train3 <- sample(1:1199, 800, replace = FALSE)
docvars(labeled.dfm, "id_numeric") <- 1:ndoc(labeled.dfm)
training_dfm <- dfm_subset(labeled.dfm, id_numeric %in% id_train3)
test_dfm <- dfm_subset(labeled.dfm, !id_numeric %in% id_train3)
nb <- textmodel_nb(training_dfm, docvars(training_dfm, "label"))
summary(nb, 10)
```

```
##
## Call:
## textmodel_nb.dfm(x = training_dfm, y = docvars(training_dfm,
##     "label"))
##
## Class Priors:
## (showing first 3 elements)
##           economy          politics sports & culture
##            0.3333            0.3333            0.3333
##
## Estimated Feature Scores:
##                       start      arab     leagu   special     envoy
## economy           0.002178 0.0003933 0.0001815 0.0006051 3.025e-05
## politics          0.001799 0.0011915 0.0006565 0.0005836 2.918e-04
## sports & culture  0.002452 0.0001486 0.0018573 0.0006686 3.715e-05
##                       crisi      kofi     annan     call    member
## economy           0.0011799 3.025e-05 3.025e-05 0.001755 0.0009379
## politics          0.0014590 1.459e-04 1.459e-04 0.002796 0.0015319
## sports & culture  0.0002972 3.715e-05 3.715e-05 0.001337 0.0006686
```

```r
dfmat_matched <- dfm_match(test_dfm, features = featnames(training_dfm))
pred_data <- predict(nb, test_dfm)
actual_class <- docvars(test_dfm, "label")
predicted_class <- pred_data
class_table <- table(actual_class, predicted_class)
confusionMatrix(class_table, mode = "everything")
```

```
## Confusion Matrix and Statistics
##
##                    predicted_class
## actual_class        economy politics sports & culture
##   economy               121        4                2
##   politics               12      154                1
##   sports & culture       15        7               83
##
## Overall Statistics
##
##                Accuracy : 0.8972
##                  95% CI : (0.8632, 0.9252)
##     No Information Rate : 0.4135
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8424
##
##  Mcnemar's Test P-Value : 0.0003567
```

```
##
## Statistics by Class:
##
##                      Class: economy Class: politics
## Sensitivity                  0.8176          0.9333
## Specificity                  0.9761          0.9444
## Pos Pred Value               0.9528          0.9222
## Neg Pred Value               0.9007          0.9526
## Precision                    0.9528          0.9222
## Recall                       0.8176          0.9333
## F1                           0.8800          0.9277
## Prevalence                   0.3709          0.4135
## Detection Rate               0.3033          0.3860
## Detection Prevalence         0.3183          0.4185
## Balanced Accuracy            0.8968          0.9389
##                      Class: sports & culture
## Sensitivity                           0.9651
## Specificity                           0.9297
## Pos Pred Value                        0.7905
## Neg Pred Value                        0.9898
## Precision                             0.7905
## Recall                                0.9651
## F1                                    0.8691
## Prevalence                            0.2155
## Detection Rate                        0.2080
## Detection Prevalence                  0.2632
## Balanced Accuracy                     0.9474
```

```
# Iteration 4

id_train4 <- sample(1:1199, 800, replace = FALSE)
docvars(labeled.dfm, "id_numeric") <- 1:ndoc(labeled.dfm)
training_dfm <- dfm_subset(labeled.dfm, id_numeric %in% id_train4)
test_dfm <- dfm_subset(labeled.dfm, !id_numeric %in% id_train4)
nb <- textmodel_nb(training_dfm, docvars(training_dfm, "label"))
summary(nb, 10)
```

```
##
## Call:
## textmodel_nb.dfm(x = training_dfm, y = docvars(training_dfm,
##     "label"))
##
## Class Priors:
## (showing first 3 elements)
##           economy          politics sports & culture
##            0.3333            0.3333            0.3333
##
## Estimated Feature Scores:
##                      start      arab     leagu   special     envoy
## economy           0.002486 0.0003894 8.986e-05 0.0005391 2.995e-05
## politics          0.001552 0.0007032 2.425e-04 0.0007759 3.152e-04
## sports & culture  0.002171 0.0001229 2.335e-03 0.0007784 4.097e-05
##                      crisi      kofi     annan     call    member
## economy           0.0009285 2.995e-05 2.995e-05 0.001498 0.0010483
## politics          0.0010912 1.212e-04 1.212e-04 0.002740 0.0015761
```

```
## sports & culture  0.0004097 4.097e-05 4.097e-05 0.001352 0.0006555
```

```
dfmat_matched <- dfm_match(test_dfm, features = featnames(training_dfm))
pred_data <- predict(nb, test_dfm)
actual_class <- docvars(test_dfm, "label")
predicted_class <- pred_data
class_table <- table(actual_class, predicted_class)
confusionMatrix(class_table, mode = "everything")
```

```
## Confusion Matrix and Statistics
##
##                    predicted_class
## actual_class        economy politics sports & culture
##   economy               109        4                 5
##   politics                6      158                 1
##   sports & culture       21       12                83
##
## Overall Statistics
##
##                Accuracy : 0.8772
##                  95% CI : (0.8409, 0.9077)
##     No Information Rate : 0.4361
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8122
##
##  Mcnemar's Test P-Value : 0.00021
##
## Statistics by Class:
##
##                      Class: economy Class: politics
## Sensitivity                  0.8015          0.9080
## Specificity                  0.9658          0.9689
## Pos Pred Value               0.9237          0.9576
## Neg Pred Value               0.9039          0.9316
## Precision                    0.9237          0.9576
## Recall                       0.8015          0.9080
## F1                           0.8583          0.9322
## Prevalence                   0.3409          0.4361
## Detection Rate               0.2732          0.3960
## Detection Prevalence         0.2957          0.4135
## Balanced Accuracy            0.8836          0.9385
##                      Class: sports & culture
## Sensitivity                           0.9326
## Specificity                           0.8935
## Pos Pred Value                        0.7155
## Neg Pred Value                        0.9788
## Precision                             0.7155
## Recall                                0.9326
## F1                                    0.8098
## Prevalence                            0.2231
## Detection Rate                        0.2080
## Detection Prevalence                  0.2907
## Balanced Accuracy                     0.9131
```

```r
# Iteration 5

id_train5 <- sample(1:1199, 800, replace = FALSE)
docvars(labeled.dfm, "id_numeric") <- 1:ndoc(labeled.dfm)
training_dfm <- dfm_subset(labeled.dfm, id_numeric %in% id_train5)
test_dfm <- dfm_subset(labeled.dfm, !id_numeric %in% id_train5)
nb <- textmodel_nb(training_dfm, docvars(training_dfm, "label"))
summary(nb, 10)
```

```
##
## Call:
## textmodel_nb.dfm(x = training_dfm, y = docvars(training_dfm,
##     "label"))
##
## Class Priors:
## (showing first 3 elements)
##           economy          politics sports & culture
##            0.3333            0.3333            0.3333
##
## Estimated Feature Scores:
##                       start      arab     leagu   special     envoy
## economy           0.002566 0.0002782 0.0001236 0.0004327 3.091e-05
## politics          0.001730 0.0010232 0.0005847 0.0007553 3.167e-04
## sports & culture  0.002489 0.0001509 0.0027534 0.0007921 3.772e-05
##                       crisi      kofi     annan     call    member
## economy           0.0012982 3.091e-05 3.091e-05 0.001700 0.0008964
## politics          0.0010476 1.218e-04 1.218e-04 0.002509 0.0015349
## sports & culture  0.0003395 3.772e-05 3.772e-05 0.001546 0.0008675
```

```r
dfmat_matched <- dfm_match(test_dfm, features = featnames(training_dfm))
pred_data <- predict(nb, test_dfm)
actual_class <- docvars(test_dfm, "label")
predicted_class <- pred_data
class_table <- table(actual_class, predicted_class)
confusionMatrix(class_table, mode = "everything")
```

```
## Confusion Matrix and Statistics
##
##                    predicted_class
## actual_class        economy politics sports & culture
##   economy               115        2                10
##   politics                7      159                 2
##   sports & culture       10       10                84
##
## Overall Statistics
##
##                Accuracy : 0.8972
##                  95% CI : (0.8632, 0.9252)
##     No Information Rate : 0.4286
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8423
##
##  Mcnemar's Test P-Value : 0.04377
```

```
##
## Statistics by Class:
##
##                      Class: economy Class: politics
## Sensitivity                  0.8712          0.9298
## Specificity                  0.9551          0.9605
## Pos Pred Value               0.9055          0.9464
## Neg Pred Value               0.9375          0.9481
## Precision                    0.9055          0.9464
## Recall                       0.8712          0.9298
## F1                           0.8880          0.9381
## Prevalence                   0.3308          0.4286
## Detection Rate               0.2882          0.3985
## Detection Prevalence         0.3183          0.4211
## Balanced Accuracy            0.9131          0.9452
##                      Class: sports & culture
## Sensitivity                           0.8750
## Specificity                           0.9340
## Pos Pred Value                        0.8077
## Neg Pred Value                        0.9593
## Precision                             0.8077
## Recall                                0.8750
## F1                                    0.8400
## Prevalence                            0.2406
## Detection Rate                        0.2105
## Detection Prevalence                  0.2607
## Balanced Accuracy                     0.9045
```

From the results above, it seems that model 2 is our best fit, so that's the one we are going to use to predict the topics in our larger dataset of 2012 articles. Therefore, we need to get the model fitted again. Fortuntely, because we used `set.seed()` we can reproduce the findings.

```
# Reproduce the findings of Iteration 2, to predict labels of the full dataset
training_dfm <- dfm_subset(labeled.dfm, id_numeric %in% id_train2)
test_dfm <- dfm_subset(labeled.dfm, !id_numeric %in% id_train2)
nb <- textmodel_nb(training_dfm, docvars(training_dfm, "label"))
```

I do not need to re-run all the metrics again, because all I am interested in is the "nb" object which has all the probabilites that I need to feed into the `predict` function below.

So, we are ready to go back to the dataframe that we created at the very beginning with all the 20,000+ articles that we haven't labeled, but want to see labeled. We are going to use the probabilities in nb to get labels.

```
# We match the features in the NB fitted model and in our dataset of interest
dfmat_matched <- dfm_match(cctv.dfm, features = featnames(training_dfm))

# We fit the model
tmod <- predict(nb, dfmat_matched)

tmod.labels <- as.data.frame(tmod)
colnames(tmod.labels) <- "labels"

head(tmod.labels, 10)
```

```
##                  labels
## text1           politics
```

```
## text2            economy
## text3            economy
## text4           politics
## text5  sports & culture
## text6  sports & culture
## text7            economy
## text8            economy
## text9            economy
## text10           economy
```

```
table(tmod.labels$labels)/length(tmod) # Proportion of topics in our target df
```

```
##
##           economy          politics sports & culture
##         0.4390263         0.2014683         0.3595054
```

The results we find here would need to be validated by hand coding a sample of these articles and making sure that the labels actually match the content of the article.