The [national library of Luxembourg](https://data.bnl.lu/data/historical-newspapers/) published  
some very interesting data sets; scans of historical newspapers! There are several data sets that  
you can download, from 250mb up to 257gb. I decided to take a look at the 32gb “ML Starter Pack”.  
It contains high quality scans of one year of the *L’indépendence Luxembourgeoise* (Luxembourguish  
independence) from the year 1877. To make life easier to data scientists, the national library  
also included ALTO and METS files (which is a XML schema that is used to describe the layout and  
contents of physical text sources, such as pages of a book or newspaper) which can be easily parsed  
by R.

*L’indépendence Luxembourgeoise* is quite interesting in that it is a Luxembourguish newspaper written  
in French. Luxembourg always had 3 languages that were used in different situations, French, German  
and Luxembourguish. Luxembourguish is the language people used (and still use) for day to day life  
and to speak to their baker.  
Historically however, it was not used for the press or in politics. Instead it was German that  
was used for the press (or so I thought) and French in politics (only in  
[1984](http://legilux.public.lu/eli/etat/leg/loi/1984/02/24/n1/jo) was Luxembourguish made  
an official Language of Luxembourg).  
It turns out however that *L’indépendence Luxembourgeoise*, a daily newspaper that does not exist  
anymore, was in French. This piqued my interest, and it also made analysis easier, for 2 reasons:  
I first started with the *Luxemburger Wort* (Luxembourg’s Word I guess would be a translation), which  
still exists today, but which is in German. And at that time, German was written using the Fraktur  
font, which makes it barely readable. Look at the alphabet in Fraktur:

𝕬 𝕭 𝕮 𝕯 𝕰 𝕱 𝕲 𝕳 𝕴 𝕵 𝕶 𝕷 𝕸 𝕹 𝕺 𝕻 𝕼 𝕽 𝕾 𝕿 𝖀 𝖁 𝖂 𝖃 𝖄 𝖅

𝖆 𝖇 𝖈 𝖉 𝖊 𝖋 𝖌 𝖍 𝖎 𝖏 𝖐 𝖑 𝖒 𝖓 𝖔 𝖕 𝖖 𝖗 𝖘 𝖙 𝖚 𝖛 𝖜 𝖝 𝖞 𝖟

It’s not like German is already hard enough, they had to invent the least readable font ever to write  
German in, to make extra sure it would be hell to decipher.

So basically I couldn’t be bothered to try to read a German newspaper in Fraktur. That’s when I noticed  
the *L’indépendence Luxembourgeoise*… A Luxembourguish newspaper? Written in French? Sounds  
interesting.

And oh boy. Interesting it was.

19th century newspapers articles were something else. There’s this article for instance:

For those of you that do not read French, this article relates that in France, the ministry of  
justice required priests to include prayers on the Sunday that follows the start of the new season  
of parliamentary discussions, in order for God to provide senators his help.

There this gem too:

This article presents the tallest soldier of the German army, called Emhke, and nominated by the  
German Emperor himself to accompany him during his visit to Palestine. Emhke was 2.08 meters tall  
and weighted 236 pounds (apparently at the time Luxembourg was not fully sold on the metric system).

Anyway, I decided to take a look at ads. The last paper of this 4 page newspaper always contained  
ads and other announcements. For example, there’s this ad for a pharmacy:

that sells tea, and mineral water. Yes, tea and mineral water. In a pharmacy. Or this one:

which is literally upside down in the newspaper (the one from the 10th of April 1877). I don’t  
know if it’s a mistake or if it’s a marketing ploy, but it did catch my attention, 140 years later,  
so *bravo*. This is an announcement made by a shop owner that wants to sell all his merchandise  
for cheap, perhaps to make space for new stuff coming in?

So I decided brush up on my natural language processing skills with R and do topic modeling on these ads.  
The challenge here is that a single document, the 4th page of the newspaper, contains a lot of ads.  
So it will probably be difficult to clearly isolate topics. But let’s try nonetheless.  
First of all, let’s load all the .xml files that contain the data. These files look like this:

I’m interested in the “CONTENT” tag, which contains the words. Let’s first get that into R.

Load the packages, and the files:

library(tidyverse)

library(tidytext)

library(topicmodels)

library(brotools)

ad\_pages <- str\_match(list.files(path = "./", all.files = TRUE, recursive = TRUE), ".\*4-alto.xml") %>%

discard(is.na)

I save the path of all the pages at once into the ad\_pages variables. To understand how and why  
this works, you must take a look at the hierarchy of the folder:

Inside each of these folder, there is a text folder, and inside this folder there are the .xml  
files. Because this structure is bit complex, I use the list.files() function with the  
all.files and recursive argument set to TRUE which allow me to dig deep into the folder  
structure and list every single file. I am only interested into the 4th page though, so that’s why  
I use str\_match() to only keep the 4th page using the ".\*4-alto.xml" regular expression. This  
is the right regular expression, because the files are named like so:

1877-12-29\_01-00004-alto.xml

So in the end, ad\_pages is a list of all the paths to these files. I then write a function  
to extract the contents of the “CONTENT” tag. Here is the function.

get\_words <- function(page\_path){

page <- read\_file(page\_path)

page\_name <- str\_extract(page\_path, "1.\*(?=-0000)")

page %>%

str\_split("\n", simplify = TRUE) %>%

keep(str\_detect(., "CONTENT")) %>%

str\_extract("(?<=CONTENT)(.\*?)(?=WC)") %>%

discard(is.na) %>%

str\_extract("[:alpha:]+") %>%

tolower %>%

as\_tibble %>%

rename(tokens = value) %>%

mutate(page = page\_name)

}

This function takes the path to a page as argument, and returns a tibble with the two columns: one  
containing the words, which I called tokens and the second the name of the document this word  
was found. The difficult part is str\_extract("(?<=CONTENT)(.\*?)(?=WC)")  
which is were the words inside the “CONTENT” tag get extracted.

I then map this function to all the pages, and get a nice tibble with all the words:

ad\_words <- map\_dfr(ad\_pages, get\_words)

ad\_words

## # A tibble: 1,114,662 x 2

## tokens page

##

## 1 afin 1877-01-05\_01/text/1877-01-05\_01

## 2 de 1877-01-05\_01/text/1877-01-05\_01

## 3 mettre 1877-01-05\_01/text/1877-01-05\_01

## 4 mes 1877-01-05\_01/text/1877-01-05\_01

## 5 honorables 1877-01-05\_01/text/1877-01-05\_01

## 6 clients 1877-01-05\_01/text/1877-01-05\_01

## 7 à 1877-01-05\_01/text/1877-01-05\_01

## 8 même 1877-01-05\_01/text/1877-01-05\_01

## 9 d 1877-01-05\_01/text/1877-01-05\_01

## 10 avantages 1877-01-05\_01/text/1877-01-05\_01

## # ... with 1,114,652 more rows

I then do some further cleaning, removing stop words (French and German, because there are some  
ads in German) and a bunch of garbage characters and words, which are probably when the OCR failed.  
I also remove some German words from the few German ads that are in the paper, because they have  
a very high tf-idf (I’ll explain below what that is).  
I also remove very common words in ads that were just like stopwords. Every ad of a shop mentioned their  
clients with *honorable clientèle*, or used the word *vente*, and so on. This is what you see below  
in the very long calls to str\_remove\_all. I also compute the tf\_idf and I am grateful to

It’s in French though, but the idea of the blog post is to present topic modeling with Wikipedia  
articles. You can also read the section on tf-idf from the Text Mining with R ebook.  
tf-idf gives a measure of how common words are. Very common words, like stopwords, have a tf-idf  
of 0. So I use this to further remove very common words, by only keeping words with a tf-idf  
greater than 0.01. This is why I manually remove garbage words and German words below, because they  
are so uncommon that they have a very high tf-idf and mess up the rest of the analysis. To find these words  
I had to go back and forth between the tibble of cleaned words and my code, and manually add all  
these exceptions. It took some time, but definitely made the results of the next steps better.  
I then use cast\_dtm to cast the tibble into a DocumentTermMatrix object, which  
is needed for the LDA() function that does the topic modeling:

stopwords\_fr <- read\_csv("https://raw.githubusercontent.com/stopwords-iso/stopwords-fr/master/stopwords-fr.txt",

col\_names = FALSE)

## Parsed with column specification:

## cols(

## X1 = col\_character()

## )

stopwords\_de <- read\_csv("https://raw.githubusercontent.com/stopwords-iso/stopwords-de/master/stopwords-de.txt",

col\_names = FALSE)

## Parsed with column specification:

## cols(

## X1 = col\_character()

## )

## Warning: 1 parsing failure.

## row col expected actual file

## 157 -- 1 columns 2 columns 'https://raw.githubusercontent.com/stopwords-iso/stopwords-de/master/stopwords-de.txt'

ad\_words2 <- ad\_words %>%

filter(!is.na(tokens)) %>%

mutate(tokens = str\_remove\_all(tokens,

'[|\\|!|"|#|$|%|&|\\\*|+|,|-|.|/|:|;|<|=|>|?|@|^|\_|`|’|\'|‘|(|)|\\||~|=|]|°|<|>|«|»|\\d{1,100}|©|®|•|—|„|“|-|¦\\\\|”')) %>%

mutate(tokens = str\_remove\_all(tokens,

"j'|j’|m’|m'|n’|n'|c’|c'|qu’|qu'|s’|s'|t’|t'|l’|l'|d’|d'|luxembourg|honneur|rue|prix|maison|frs|ber|adresser|unb|mois|vente|informer|sann|neben|rbudj|artringen|salz|eingetragen|ort|ftofjenb|groifdjen|ort|boch|chem|jahrgang|uoa|genannt|neuwahl|wechsel|sittroe|yerlorenkost|beichsmark|tttr|slpril|ofto|rbudj|felben|acferftücf|etr|eft|sbege|incl|estce|bes|franzosengrund|qne|nne|mme|qni|faire|id|kil")) %>%

anti\_join(stopwords\_de, by = c("tokens" = "X1")) %>%

filter(!str\_detect(tokens, "§")) %>%

mutate(tokens = ifelse(tokens == "inédite", "inédit", tokens)) %>%

filter(tokens != "") %>%

anti\_join(stopwords\_fr, by = c("tokens" = "X1")) %>%

count(page, tokens) %>%

bind\_tf\_idf(tokens, page, n) %>%

arrange(desc(tf\_idf))

dtm\_long <- ad\_words2 %>%

filter(tf\_idf > 0.01) %>%

cast\_dtm(page, tokens, n)

To read more details on this, I suggest you take a look at the following section of the.

I choose to model 10 topics (k = 10), and set the alpha parameter to 5. This hyperparamater controls how  
many topics are present in one document. Since my ads are all in one page (one document), I  
increased it. Let’s fit the model, and plot the results:

lda\_model\_long <- LDA(dtm\_long, k = 10, control = list(alpha = 5))

I plot the per-topic-per-word probabilities, the “beta” from the model and plot the 5 words that  
contribute the most to each topic:

result <- tidy(lda\_model\_long, "beta")

result %>%

group\_by(topic) %>%

top\_n(5, beta) %>%

ungroup() %>%

arrange(topic, -beta) %>%

mutate(term = reorder(term, beta)) %>%

ggplot(aes(term, beta, fill = factor(topic))) +

geom\_col(show.legend = FALSE) +

facet\_wrap(~ topic, scales = "free") +

coord\_flip() +

theme\_blog()

So some topics seem clear to me, other not at all. For example topic 4 seems to be about shoes made  
out of leather. The word semelle, sole, also appears.  
Then there’s a lot of topics that reference either music, bals, or instruments.  
I guess these are ads for local music festivals, or similar events. There’s also an ad for what  
seems to be bundles of sticks, topic 3: chêne is oak, copeaux is shavings and you know  
what fagots is. The first word stère which I did not know is a unit of volume equal to one  
cubic meter. So they were likely selling  
bundle of oak sticks by the cubic meter. For the other topics, I either  
lack context or perhaps I just need to adjust k, the number of topics to model, and alpha to get better  
results. In the meantime, topic 1 is about shoes (chaussures), theatre, fuel (combustible)  
and farts (pet). Really wonder what they were selling in that shop.

In any case, this was quite an interesting project. I learned a lot about topic modeling  
and historical newspapers of my country! I do not know if I will continue exploring it myself,  
but I am really curious to see what others will do with it!