More Text Analysis with R

36-290 – Statistical Research Methodology

Week 12 Thursday – Fall 2021

Topic Modeling

Topic modeling denotes a set of unsupervised learning techniques that one can use to uncover topical structure in a set of documents, i.e., the "latent semantic structures of an extensive text body."

The sets of documents can include, e.g.,

- tweets and blog posts;
- news articles;
- essays (e.g., the *Federalist Papers*);
- etc.

There are a number of defined algorithms for topic modeling; the one that we will look at in this set of notes is *Latent Dirichlet Allocation*, or *LDA*.

Topic Modeling: the GrantForward Database

GrantForward is a searchable database of grant proposal solicitations. From this, we extract a list of solicitations (873 in all). There are many columns of information for each solicitation; here, we will focus on the solicitation `description`.

```
gf   = read.csv("http://www.stat.cmu.edu/~pfreeman/gf.csv",stringsAsFactors=FALSE)
desc = data.frame("line"=1:length(gf$description),"text"=gf$description)

substr(desc$text[1],1,80)
```

```
## [1] "Funding Opportunity Number: PAS-21-215\n\nPurpose/Research Objectives\nThere is an "
```

```
nchar(desc$text[1])
```

```
## [1] 7656
```

Can we use the words in the descriptions to define natural groupings of topics?

TidyText Pre-Processing

Here we tokenize, remove basic stopwords, generate a data frame with document-term frequency (how many times does a particular word appear in a particular document), and last, convert that `tidytext` format file to a so-called `document term matrix` (or `dtm`). A dtm is literally just an alternative means to store document-term frequency data…and we make use of this alternative because it is the format that our topic-modeling function expects.

```
suppressMessages(library(tidyverse))
suppressMessages(library(stopwords))
suppressMessages(library(tm))
library(topicmodels)
library(tidytext)

desc %>% unnest_tokens(word,text) -> desc_df

words.log = desc_df$word %in% stopwords::stopwords("en")
desc_df   = desc_df[words.log==FALSE,]

desc_df %>%
  count(line,word,sort=TRUE) %>%
  ungroup() -> desc_cts

desc_cts %>% cast_dtm(line,word,n) -> desc_dtm
desc_dtm
```

```
## <<DocumentTermMatrix (documents: 873, terms: 14013)>>
## Non-/sparse entries: 118400/12114949
## Sparsity           : 99%
## Maximal term length: 55
## Weighting          : term frequency (tf)
```

Note that a dtm can also be defined via tf-idf weighting.

Topic Modeling: Latent Dirichlet Allocation

The basic idea of LDA is that

- every document is a mixture of topics (e.g., this document is 30% topic A and 70% topic B, while this other document is 90% topic A and 10% topic B, etc.); and

- every topic is a mixture of words.

The inputs to an LDA algorithm are a document-term matrix and the number of topics K. As with K-means, there is no *a priori* heuristic for what K should be. Just try multiple values and see if you can create a story from the output! (Note that there is a concept dubbed *coherence score* that is not explored in Text Mining with R and thus will not be explored here. This can help quantify the "quality" of the topics that LDA uncovers.)

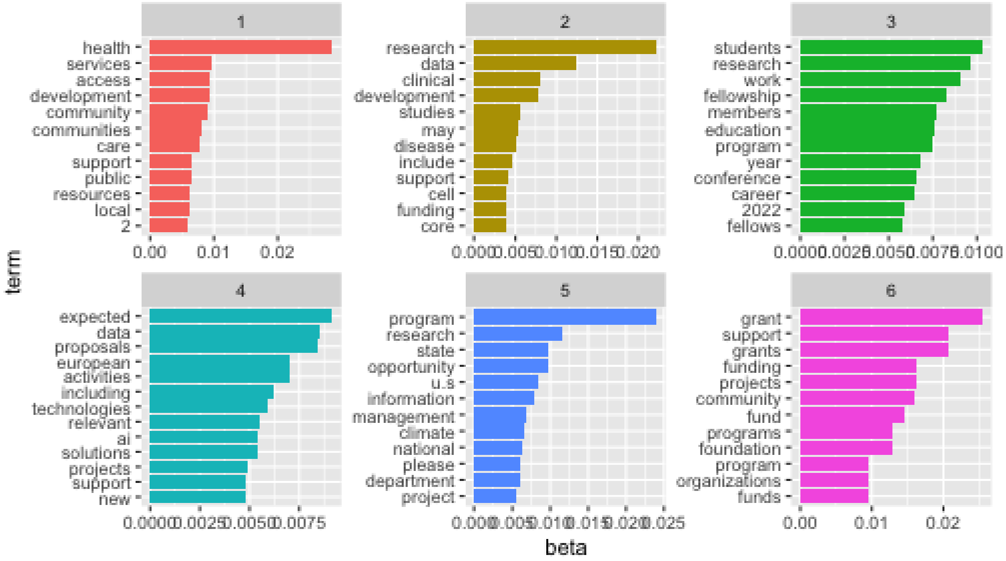
Note that the same individual word can appear in multiple topics.

For more on the mathematics of LDA, see this web page.

Topic Modeling: Latent Dirichlet Allocation

Here, let's try six groups and see if we can interpret these groups given their top terms.

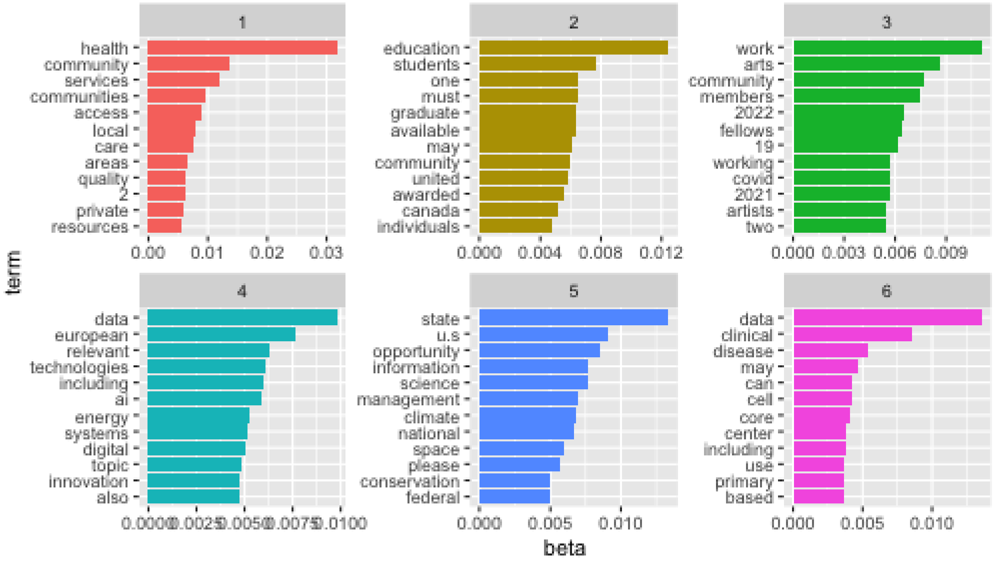
```
desc_lda = LDA(desc_dtm,k=6,method="Gibbs",
                control=list(seed=202,burnin=250,iter=250))

tidy(desc_lda,matrix="beta") %>% group_by(topic) %>%
  slice_max(beta,n=12) %>%
  ungroup() %>%
  arrange(topic,-beta) -> desc_top
desc_top %>%
  mutate(term=reorder_within(term,beta,topic)) %>%
  ggplot(aes(beta,term,fill=factor(topic))) +
  geom_col(show.legend=FALSE) +
  facet_wrap(~topic,scales="free") +
  scale_y_reordered()
```



Customizing Stopwords

What we discovered on the last slide is that topic groups can be hard to interpret when the documents contain uninformative words that are not classic stopwords. This commonly happens in text analysis. One way to get around this issue is to create a custom dictionary of stopwords, perhaps iterating until a more coherent story can be constructed from the uncovered topic groups…

```
words = desc_df$word  # the new stopword list is not necessarily complete or optimal!
new_stopwords = c("grant","proposal","funding","research","program","fellowship","year","support",
                  "conference","foundations","grants","fund","funds","department","solutions","new",
                  "projects","proposals","development","organizations","activities","studies","include",
                  "scholarship","applicants","applications","application","expected","programs",
                  "provide","foundation","project","award","awards","public")
words.log = words %in% new_stopwords
desc_df = desc_df[words.log==FALSE,]
```



Mapping Documents to Topics

A natural question that arises in LDA is "can we map particular documents to particular groups," meaning can we specify the percentages of each topic group in a document?

```
gamma = tidy(desc_lda,matrix="gamma")  # gamma: per topic per document probabilities
head(gamma,3)
```

```
## # A tibble: 3 × 3
##   document topic gamma
##   <chr>    <int> <dbl>
## 1 661          1 0.977
## 2 662          1 0.979
## 3 663          1 0.974
```

```
w = which(gamma$document==661)
gamma$gamma[w]
```

```
## [1] 0.976815398 0.003062117 0.004374453 0.005358705 0.006014873 0.004374453
```

The 661st grant solicitation is dominated by topic one, which has words like "health," "community," and "care," implying it is about public health.

Statistical Learning with DocumentTermMatrix Objects

We'll conclude by showing how one can use a document-term matrix as a data frame in a statistical learning context. Here we filter the GrantForward dataset so that it only includes those solicitations whose top-level subject areas are `Medical Sciences` and `Urbanism`. Can we train a classifier that can predict whether a given GrantForward description is to be mapped to one or the other? (Note: not shown is the processing of the new `gf` data frame into a dtm object, which proceeds identically to what we show above.)

```
gf        = read.csv("http://www.stat.cmu.edu/~pfreeman/gf.csv",stringsAsFactors=FALSE)
s         = strsplit(gf$subjects,split=":")
top.s     = sapply(s,function(x){return(x[1])})
w         = which(top.s=="Urbanism"|top.s=="Medical Sciences")
label     = factor(top.s[w])
gf        = gf[w,]
```

Statistical Learning with DocumentTermMatrix Objects

```
m = as.data.frame(as.matrix(desc_dtm))
dim(m)
```

```
## [1]  250 7631
```

Note the size of the matrix: $p \gg n$. ML methods like random forest can handle such data (albeit slowly), but logistic regression cannot handle this at all. (Lasso…yes…straight-up logistic regression…no.) So we will only keep those columns with large numbers of data across documents:

```
w = which(colSums(m)>50)
m = m[,w]
dim(m)
```

```
## [1] 250 173
```

```
set.seed(747)
m       = data.frame(m,"label"=label)
train   = sample(nrow(m),round(0.7*nrow(m)))
m.train = m[train,]
m.test  = m[-train,]
```

Statistical Learning with DocumentTermMatrix Objects

```
glm.out  = glm(label~.,data=m.train,family=binomial)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
glm.prob = predict(glm.out,newdata=m.test,type="response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == : prediction from a rank-deficient fi
## misleading
```

```
glm.pred = ifelse(glm.prob>151/250,"Urbanism","Medical Sciences")
mean(glm.pred!=m.test$label)
```

```
## [1] 0.48
```

```
table(glm.pred,m.test$label)
```

```
##
## glm.pred           Medical Sciences Urbanism
##   Medical Sciences               16       19
##   Urbanism                       17       23
```

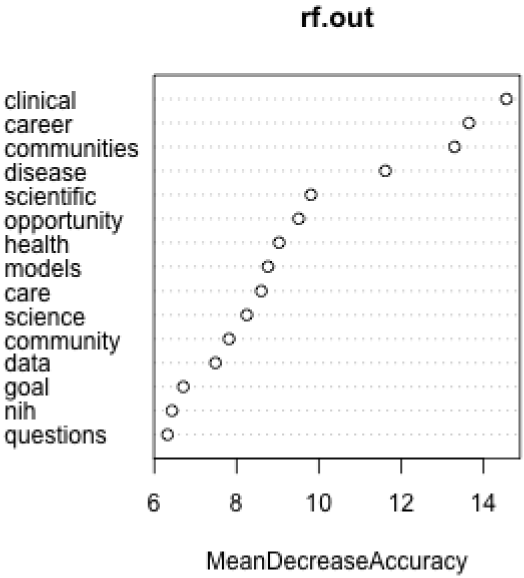
Note the warning about the algorithm not converging. This is not shocking given the number of predictor variables, and thus it is not shocking that logistic regression did worse than if we had just classified everything as `Urbanism` (misclassification rate: 39.6%).

(Also note the 151/250: `Urbanism`, with 151 rows, is both Class 1 and the dominant class. So we shift the threshold *up* towards Class 1 to try to classify correctly more Class 0 data. If `Urbanism` were Class 0, we'd shift *down* to 99/250.)

Statistical Learning with DocumentTermMatrix Objects

```
suppressMessages(library(randomForest)) ; set.seed(101)
rf.out  = randomForest(label~.,data=m.train,importance=TRUE)
rf.prob = predict(rf.out,newdata=m.test,type="prob")[,2]
rf.pred = ifelse(rf.prob>151/250,"Urbanism","Medical Sciences")
mean(rf.pred!=m.test$label) ; table(rf.pred,m.test$label) ; varImpPlot(rf.out,type=1,n.var=15)
```

```
## [1] 0.1466667

##
## rf.pred            Medical Sciences Urbanism
##   Medical Sciences               27        5
##   Urbanism                        6       37
```


rf.out
clinical
career
communities
disease
scientific
opportunity
health
models
care
science
community
data
goal
nih
questions
MeanDecreaseAccuracy