Classification, NER, LSA and LDA

NLP master course 2021-2022

Mariano Rico (mariano.rico@upm.es)

Document created on 2023-01-04

Table of contents

1 Document classification

1) Using the LMRD corpus

The LMRD (Large Movie Review Dataset) corpus contains 50,000 movie reviews in English (Mass et al. 2011). Only requires 26MB in disc and 106MB in RAM.

Download the file `data_corpus_LMRD.rds` from here and move it to your working directory (wd). Reminder: the wd will be the result of `getwd()` if you are using a `.R` file but, if you are using a `.Rmd` wd will be the directory of the `.Rmd` file.

```
library(quanteda)
data_corpus_LMRD <- readRDS("data_corpus_LMRD.rds") #It is a quanteda corpus
dfmat <- dfm(tokens(data_corpus_LMRD)) #Computes the matrix docs-feat (50.000 docs x 149.653 feats)

#The corpus has docvars
head(docvars(data_corpus_LMRD))
```

docnumber	rating	set	polarity
0	2	test	neg
10000	4	test	neg
10001	1	test	neg
10002	3	test	neg
10003	3	test	neg
10004	2	test	neg

```
#Notice that the dfm keeps the corpus docvars
head(docvars(dfmat))
```

docnumber	rating	set	polarity
0	2	test	neg
10000	4	test	neg
10001	1	test	neg
10002	3	test	neg
10003	3	test	neg
10004	2	test	neg

We can create *train* and *test* subsets:

```
dfmat_train <- dfm_subset(dfmat, set == "train") #The dfm is filtered by conditions on docvars
dfmat_test <- dfm_subset(dfmat, set == "test")
```

1.1 Creating a Naive Bayes model

You can compute a *Naive Bayes* (nb) model using `textmodel_nb()`, available in the `quanteda.textmodels` package:

```r
library("quanteda.textmodels")
#Computes a nb (Naive Bayes) model
multi <- textmodel_nb(dfmat_train,
                      dfmat_train$polarity,
                      distribution = "multinomial")
#Predicts with the test data
pred <- predict(multi, #the computed model
                newdata = dfmat_test)
```

1.2 Computing the confusion matrix

We can compute the confusion matrix for our prediction:

```r
library(caret) #The confusionMatrix() funtion is available in this package
confM <- confusionMatrix(pred, docvars(dfmat_test)$polarity)
confM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   neg   pos
##        neg 10993  3167
##        pos  1507  9333
##
##                Accuracy : 0.813
##                  95% CI : (0.8082, 0.8179)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6261
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8794
##             Specificity : 0.7466
##          Pos Pred Value : 0.7763
##          Neg Pred Value : 0.8610
##              Prevalence : 0.5000
##          Detection Rate : 0.4397
##    Detection Prevalence : 0.5664
##       Balanced Accuracy : 0.8130
##
##        'Positive' Class : neg
##
```

Notice that the output of `confusionMatrix()` has no *precision* or *recall*. Indeed, they appear but with other names: `Pos Pred Value` and `Sensitivity` respectively. You can get these values with:

```r
precision <- confM$byClass['Pos Pred Value']
precision
```

```
## Pos Pred Value
##      0.7763418
```

```
recall <- confM$byClass['Sensitivity']
recall
```

```
## Sensitivity
##     0.87944
```

2 Name Entity Recognition using collocations

We will use again the corpus created in the previous hands-on from Don Quijote chapters. Here (file Qcaps.rds) you have an R object serialized. This object is a list of strings, in which each string is a chapter from Cervantes' novel.

```
library(quanteda)
library(quanteda.textstats) #Required to use textstat_collocations()
#Load chapters and create corpus
chaps <- readRDS(file="Qcaps.rds")
texts_chaps <- unlist(chaps)
names(texts_chaps) <- paste("Cap.", 1:length(texts_chaps)) #Assigns a name to each string
corpus_chapsQ <- corpus(texts_chaps)
docvars(corpus_chapsQ, field="Chapter") <- 1:length(texts_chaps) #docvar with chapter number
corpus_chapsQ
```

```
## Corpus consisting of 126 documents and 1 docvar.
## Cap. 1 :
## "Capítulo primero. Que trata de la condición y ejercicio del ..."
##
## Cap. 2 :
## "Capítulo II. Que trata de la primera salida que de su tierra..."
##
## Cap. 3 :
## "Capítulo III. Donde se cuenta la graciosa manera que tuvo do..."
##
## Cap. 4 :
## "Capítulo IV. De lo que le sucedió a nuestro caballero cuando..."
##
## Cap. 5 :
## "Capítulo V. Donde se prosigue la narración de la desgracia d..."
##
## Cap. 6 :
## "Capítulo VI. Del donoso y grande escrutinio que el cura y el..."
##
## [ reached max_ndoc ... 120 more documents ]
```

In order to detect proper names in the corpus we will use `textstat_collocations()` using the tokens, removing puctuation and symbols in `tokens()`. We also remove *stop words* with `tokens_remove()`. IMPORTANT!: we use `textstat_collocations()` with `padding = TRUE` to keep the original position of tokens. Also we will assign `padding = TRUE` in `tokens()` and in `tokens_remove()`.

```
#Tokenization
toks1 <- tokens(corpus_chapsQ,
                remove_punct =   TRUE,
                remove_symbols = TRUE,
                padding =        TRUE)
toks2 <- tokens_remove(toks1,
                       stopwords("es"), #Spanish stop words (by quanteda)
                       padding = TRUE)
```

We select the tokens that begin with upper case (proper names, locations or companies) using the function `tokens_select()`:

```
toks_ucase <- tokens_select(toks2,
                            pattern = "^[A-Z]",      #Words starting with upper case
                            valuetype = "regex",     #Use a regex
                            case_insensitive = FALSE,#Case sensitive
                            padding = TRUE)          #Uses padding
```

Show collocations of 2 uppercase-starting words:

```
cols_name_2 <- textstat_collocations(toks_ucase,
                                     min_count = 4,   #The collocation must appear at least
                                                      # this number of times.
                                     size= 2,         #Number of words in the collocation
                                     tolower = FALSE) #WARN! By default is TRUE
head(cols_name_2, 20) #Shows the first 20
```

collocation	count	count_nested	length	lambda	z
Sancho Panza	273	0	2	7.151235	46.16539
Sansón Carrasco	33	0	2	9.447923	31.53760
Teresa Panza	31	0	2	6.736724	29.29946
Don Quijote	79	0	2	6.872433	25.93717
Hamete Benengeli	14	0	2	11.013277	19.06384
San Pedro	9	0	2	7.589193	18.95570
Santa Hermandad	18	0	2	11.644789	18.68132
San Juan	6	0	2	8.356149	17.39355
Pedro Recio	13	0	2	9.593319	17.30428
Cide Hamete	35	0	2	14.400687	17.04390
Gran Turco	4	0	2	9.829155	14.87429
Santa Iglesia	4	0	2	7.942861	14.50184
Blanca Luna	17	0	2	14.204224	14.13705
Triste Figura	47	0	2	16.049987	13.79123
Cervantes Saavedra	6	0	2	12.761860	12.74700
Doce Pares	6	0	2	13.213850	12.56339
Gran Capitán	6	0	2	11.792440	12.50361
Aldonza Lorenzo	6	0	2	11.267893	12.14109
Don Antonio	4	0	2	5.857775	11.73209
Válame Dios	17	0	2	9.250307	10.85772

And the collocations with 3 words:

```
cols_name_3 <- textstat_collocations(toks_ucase,
                                     min_count = 4,

                                     size= 3,
                                     tolower = FALSE)
head(cols_name_3, 20)
```

collocation	count	count_nested	length	lambda	z
Iglesia Católica Romana	4	0	3	-7.78231	-2.428298
Cide Hamete Benengeli	13	0	3	-11.74715	-7.014964

3 Latent Semantic Analysis (LSA)

3.1 Basic example

The code of this example can be found here.

In order to create LSA model we need a dfm. For example:

```
library(quanteda)
txt <- c(d1 = "Shipment of gold damaged in a fire",
         d2 = "Delivery of silver arrived in a silver truck",
         d3 = "Shipment of gold arrived in a truck" )

mydfm <- dfm(tokens(txt))
print(mydfm, max_nfeat=11)#By default only shows up to 10 features
```

```
## Document-feature matrix of: 3 documents, 11 features (36.36% sparse) and 0 docvars.
##     features
## docs shipment of gold damaged in a fire delivery silver arrived truck
##   d1        1  1    1       1  1 1    1        0      0       0     0
##   d2        0  1    0       0  1 1    0        1      2       1     1
##   d3        1  1    1       0  1 1    0        0      0       1     1
```

We need the `quanteda.textmodels` package to use the `textmodel_lsa()` function:

```
library("quanteda.textmodels")
mylsa <- textmodel_lsa(mydfm)
```

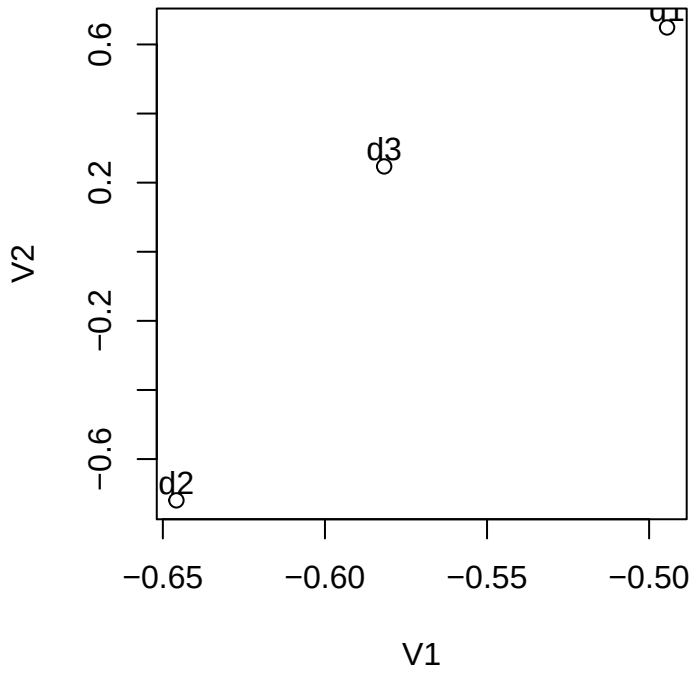
From the model we obtain the *coordinates* of the documents in a smaller space. For example in a two-dimensional space, these would be the coordinates of the documents:

```
mylsa$docs[, 1:2]
```

```
##          [,1]       [,2]
## d1 -0.4944666  0.6491758
## d2 -0.6458224 -0.7194469
## d3 -0.5817355  0.2469149
```

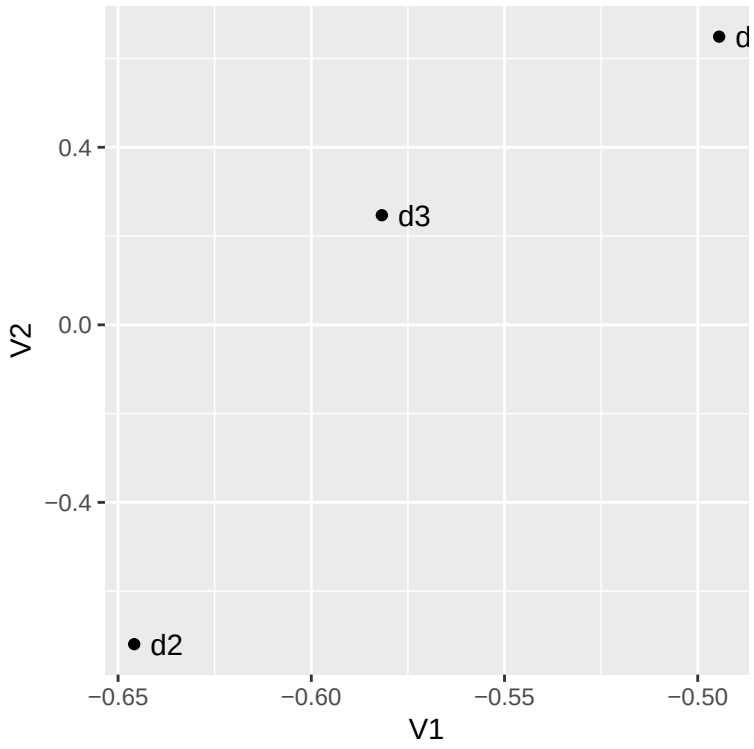
We can create a simple graph with:

```
df <- as.data.frame(mylsa$docs[, 1:2])
plot(df)
text(df$V1, 0.05+df$V2, rownames(df))
```



Or someone more advanced:

```
library(ggplot2)
ggplot(df, aes(x=V1, y=V2, label=rownames(df))) +
  geom_point() +
  geom_text(hjust=-0.5)
```



This model can be applied to a new text. First, we have to know which featured of the new document already exist in the dfm of our model. We will use the `dfm_match()` and `featnames()` functions:

```
dfm_new_doc <- dfm(c(d4 = "gold silver truck"))
new_dfm <- dfm_match(dfm_new_doc,
                     features = featnames(mydfm))

print(new_dfm, max_nfeat=11)
```

```
## Document-feature matrix of: 1 document, 11 features (72.73% sparse) and 0 docvars.
##     features
## docs shipment of gold damaged in a fire delivery silver arrived truck
##   d4        0  0    1       0  0 0    0        0      1       0     1
```

Now we can use `predict()` to obtain the coordinates of the new text/document:

```
pred_new <- predict(mylsa, newdata = new_dfm)
pred_new$docs_newspace[, 1:2]
```

```
## [1] -0.2140026 -0.1820571
```

3.2 Sonnets of the Spanish Golden Age

We will use the copus of sonnets from here (fichero `SonetoMasters4-56.rds`). It is a `quanteda` corpus with several docvars: `autor` (author, in Spanish), `num` y `file`. Each document in the corpus is a sonnet. There are 56 sonnets by 4 authors os the Spanish Golden Age (Siglo de Oro, in Spanish).

We can usa LSA to identify text *clusters* for each author.

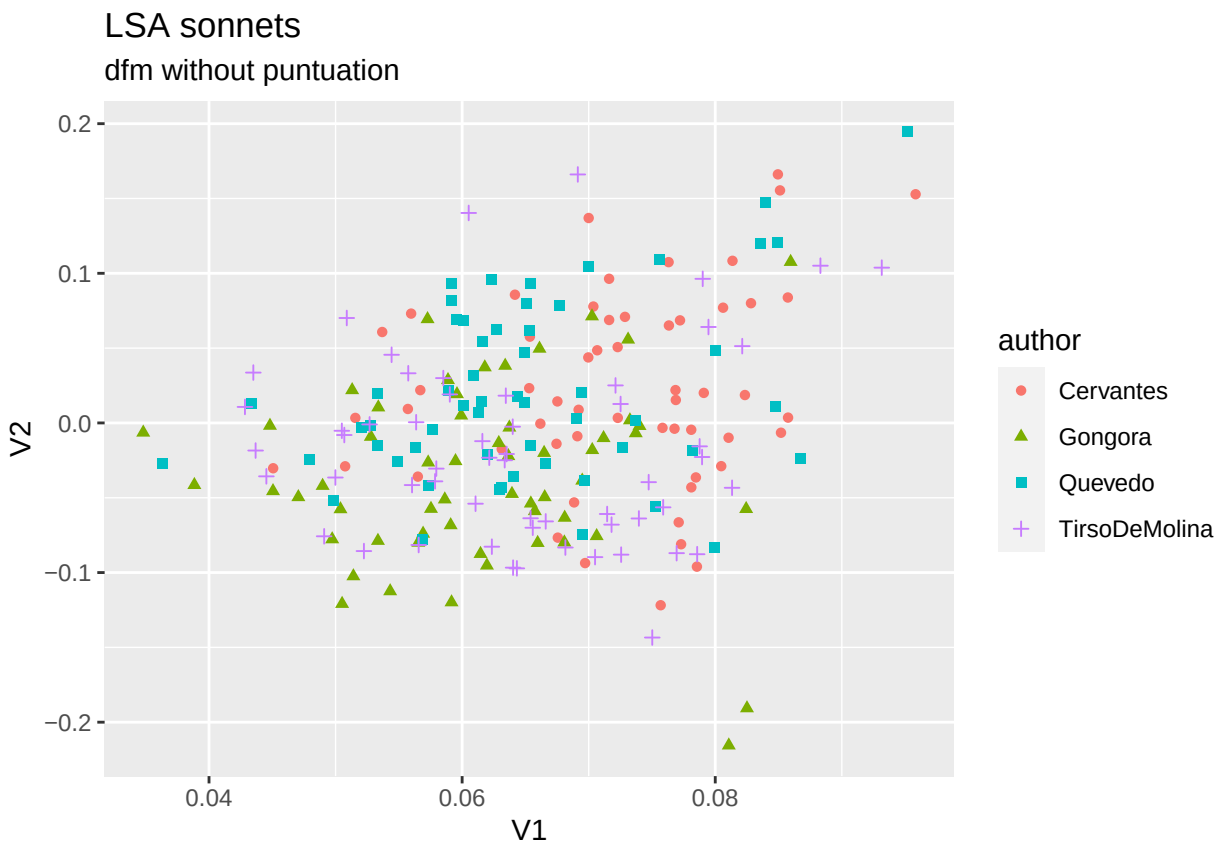
If we use all the features of the corpus, just removing punctuation, how many *features* do we have in the dfm?

```
library(quanteda)
corpusSonetos <- readRDS("SonetoMasters4-56.rds")
dfm_sonnets <- dfm(tokens(as.character(corpusSonetos),
                          remove_punct = TRUE)
                   )
nfeat(dfm_sonnets)
```

```
## [1] 5173
```

```
library("quanteda.textmodels")
mylsa <- textmodel_lsa(dfm_sonnets)

library(ggplot2)
df <- as.data.frame(mylsa$docs[, 1:2]) #In 2D
df$author <- corpusSonetos$autor
gg_ini <- ggplot(df, aes(x=V1,
                 y=V2,
                 shape=author,
                 col=author)) +
            geom_point() +
            labs(title="LSA sonnets",
            subtitle = "dfm without puntuation")
gg_ini
```



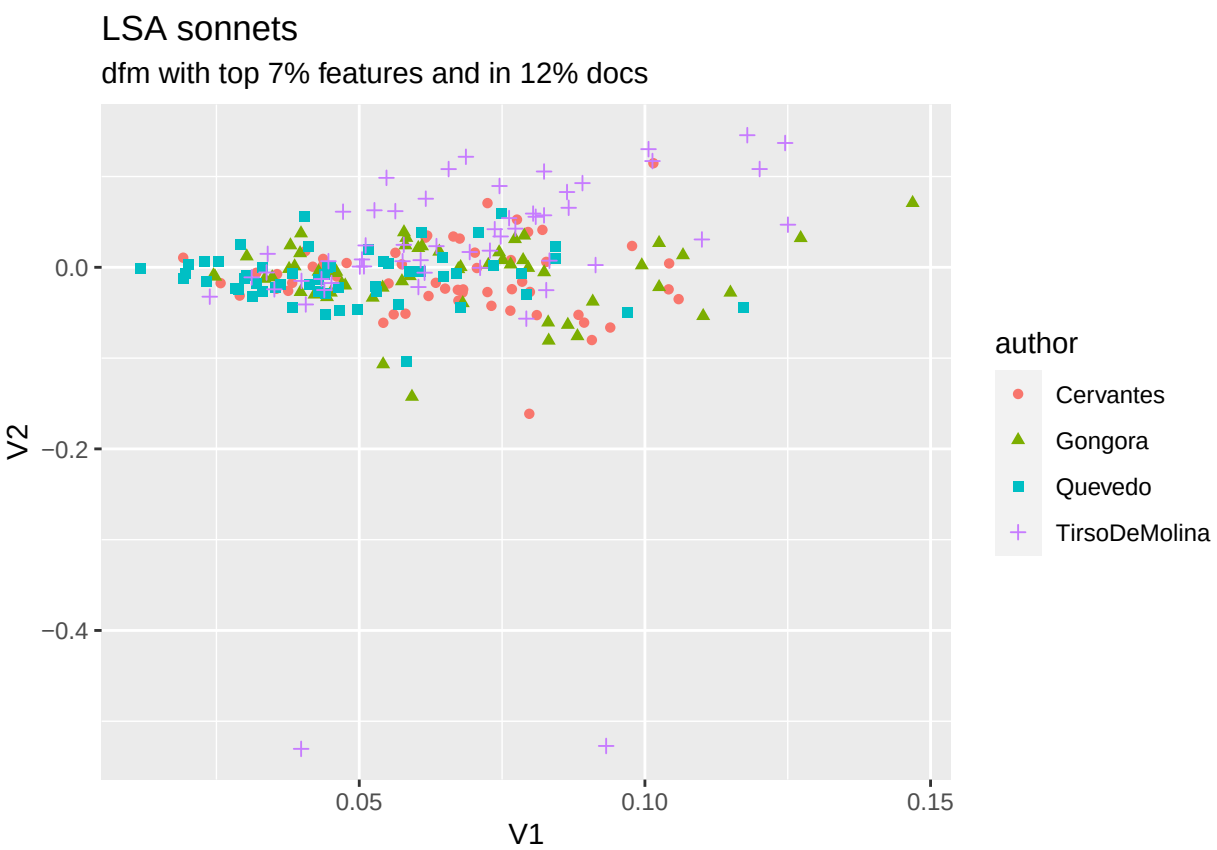
If we reduce the number of *features*, specifically keepting only the top 7% of the features appearing in less than 12% of the documents, we can identify some author better that the rest:

```
dfmat_top <- dfm_sonnets %>%
  dfm_trim(min_termfreq = 0.93, termfreq_type = "quantile",
           max_docfreq  = 0.12, docfreq_type = "prop")
nfeat(dfmat_top)
```

```
## [1] 302
```

```
mylsa <- textmodel_lsa(dfmat_top)

df <- as.data.frame(mylsa$docs[, 1:2]) #In 2D
df$author <- corpusSonetos$autor
ggplot(df, aes(x=V1,
               y=V2,
               label=rownames(df),
               shape=author,
               col=author)) +
geom_point() +
labs(title="LSA sonnets",
    subtitle = "dfm with top 7% features and in 12% docs")
```



The author `TirsoDeMolina` has a higher occurrency in the upper part of the graph.

To whom could we attribute the following sonnet? Look for the closest sonnet in the corpus.

```
Sabido he por mi mal adonde llega
la cruda fuerza de un notorio engaño,
y como amor procura, con mi daño,
```

```
darme la vida que el temor me niega.

Mi alma de las carnes se despega,
siguiendo aquella que, por hado extraño,
le tiene puesta en pena, en mal tamaño,
que el bien la turba y el dolor sosiega.

Si vivo, vivo en fe de la esperanza,
que, aunque es pequeña y débil, se sustenta
siendo a la fuerza de mi amor asida.

¡O firme comenzar, frágil mudanza,
amarga suma de una dulce cuenta,
cómo acabáis por términos la vida!
```

We convert it to a string:

```
lines <- c(
            "Sabido he por mi mal adonde llega",
                "la cruda fuerza de un notorio engaño,",
                "y como amor procura, con mi daño,",
                "darme la vida que el temor me niega.",

              "Mi alma de las carnes se despega,",
                "siguiendo aquella que, por hado extraño,",
                "le tiene puesta en pena, en mal tamaño,",
                "que el bien la turba y el dolor sosiega.",

              "Si vivo, vivo en fe de la esperanza,",
                "que, aunque es pequeña y débil, se sustenta",
                "siendo a la fuerza de mi amor asida.",

              "¡O firme comenzar, frágil mudanza,",
                "amarga suma de una dulce cuenta,",
                "cómo acabáis por términos la vida!"
            )
new_doc <- paste(lines, collapse = " ")
```

We calculate its dfm "on the basis of the dfm of the sonnets":

```
dfm_new_doc <- dfm(tokens(c(d57 = new_doc),
                          remove_punct = TRUE),
                   )
new_dfm <- dfm_match(dfm_new_doc,
                     features = featnames(dfm_sonnets))
```

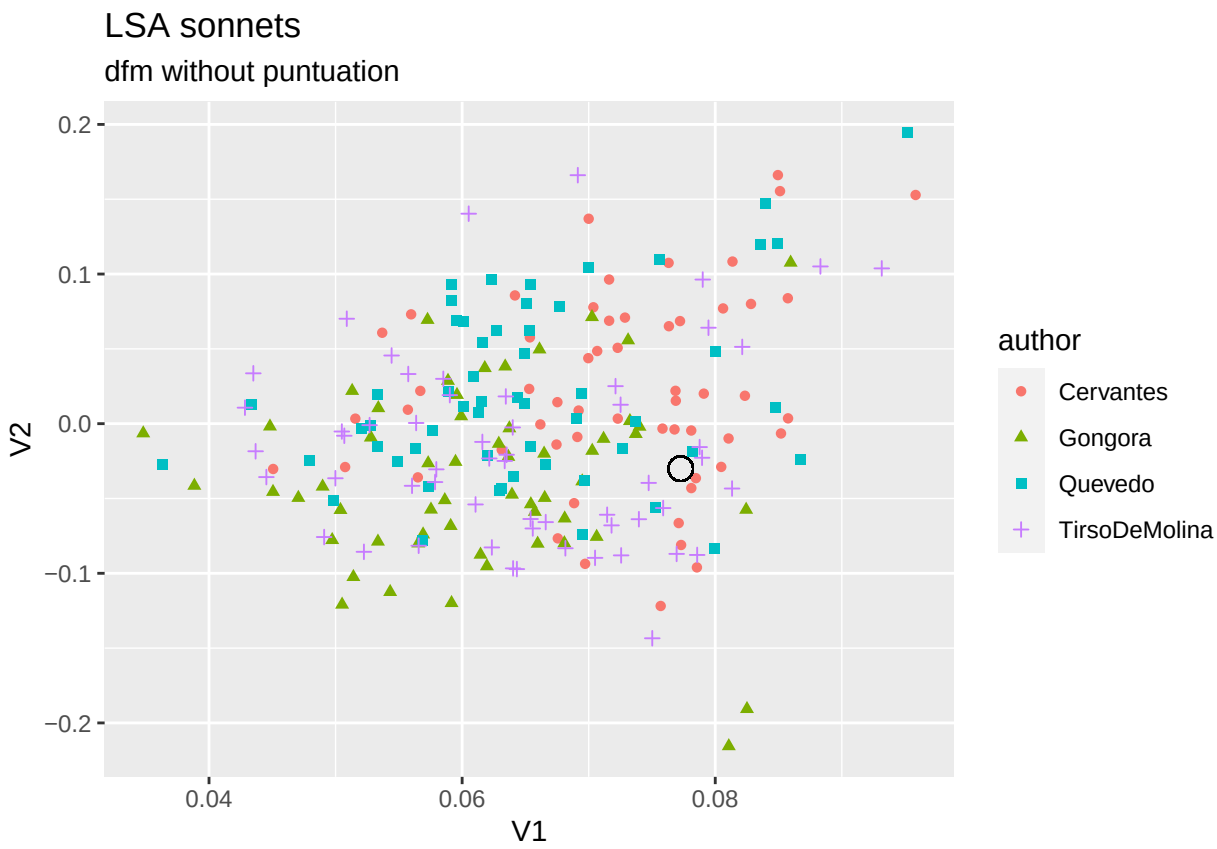
We calculate the coordinates of the new document and paint it:

```
lsa_model <- textmodel_lsa(dfm_sonnets)
pred_new <- predict(lsa_model, newdata = new_dfm)
coords <- pred_new$docs_newspace[, 1:2]

coords
```

```
## [1]  0.07727149 -0.03008982
```

```
#Reuse the initial ggplot and plot on top of it
gg_ini + geom_point(x=coords[1],
                    y=coords[2],
                    size=4,      #big
                    col="black", #color
                    shape=1      #circle
                   )
```



The closest is:

```
#The sonnet coords
df <- as.data.frame(lsa_model$docs[, 1:2]) #In 2D
#And its author
df$author <- corpusSonetos$autor

#Function to calculate the Euclidean distance of two vectors/points
dist_euclidean <- function(a, b) {sqrt(sum((a - b)^2))}

#Calculates the distance between my point and all the others
dist <- c()
for (i in 1:nrow(df)){
    dist[i] <- dist_euclidean(coords, c(df$V1[i], df$V2[i]))
}
```

```r
#The minimum
df$author[which.min(dist)]
```

```
## [1] Cervantes
## Levels: Cervantes Gongora Quevedo TirsoDeMolina
```

Then, the closest author is Cervantes. The closer, ordered, are:

```r
ordered <- sort(dist, index.return=TRUE) #Notice that using sort with index.return=TRUE
                                         # we get the values in $x and the indices in $ix
n <- 5 #The 5 closer
#The authors
head(df$author[ordered$ix], n)
```

```
## [1] Cervantes     Cervantes     TirsoDeMolina TirsoDeMolina Quevedo
## Levels: Cervantes Gongora Quevedo TirsoDeMolina
```

```r
#The distances
head(ordered$x,  n)
```

```
## [1] 0.003420935 0.006437733 0.007502904 0.009825320 0.011108625
```

4 Linear discriminant analysis (LDA)

Example adapted from here.

We will work with the `data_corpus_guardian` news corpus, which contains 6000 articles in English from The Guardian newspaper written between 2012 and 2016. It occupies 10.8 MB on disk and 29.2 MB in memory.

```r
#The function download() is in the package quanteda.corpora
library(quanteda.corpora)
corp_news <- download("data_corpus_guardian")
```

You can see that one of the *docvars* in this corpus is `date`, y that the class of these objects is `Date`:

```r
corp_news$date[1]        #It seems a string because the output of printf(someDate) is a string
```

```
## [1] "2016-02-26"
```

```r
class(corp_news$date[1]) #But indeed it is a Date class
```

```
## [1] "Date"
```

With R we manage dates and extract its day, month abd year with:

```r
fechaDate <- corp_news$date[1]

as.numeric(format(fechaDate, '%d'))
```

```
## [1] 26
```

```
as.numeric(format(fechaDate, '%m'))
```

```
## [1] 2
```

```
as.numeric(format(fechaDate, '%Y'))
```

```
## [1] 2016
```

We can filter our corpus (using `corpus_subset()`) to keep the articles written in 2016:

```
corp_news_2016 <- corpus_subset(corp_news, as.numeric(format(date, '%Y')) == 2016)
ndoc(corp_news_2016)
```

```
## [1] 1959
```

The function that created the LDA model needs a dfm not too big. Therefore, we start removing stopwords (and some other patterns) with `tokens_remove()`:

```
toks_news <- tokens(corp_news_2016,
                    remove_punct   = TRUE,  #Default is FALSE
                    remove_numbers = TRUE,  #Default is FALSE
                    remove_symbol  = TRUE)  #Default is FALSE
toks_news <- tokens_remove(toks_news,
                           pattern = c(stopwords("en"), "*-time", "updated-*", "gmt", "bst"))
```

The current number of *features* is:

```
nfeat(dfm(toks_news))
```

```
## [1] 55636
```

Still it is too big to compute the LDA. We can filter the dfm to keep, for instance, the top 1% of the *features* more frequent that appear in less than 10% of the documents. We use `dfm_trim()`:

```
dfmat_news <- dfm(toks_news) %>%
              dfm_trim(min_termfreq = 0.99, termfreq_type = "quantile",
                       max_docfreq = 0.10,  docfreq_type = "prop")
nfeat(dfmat_news)
```

```
## [1] 76
```

We can know which are these *features* and the number of occurrences of each of them with `topfeatures()`:

```
topfeatures(dfmat_news, nfeat(dfmat_news))
```

```
##     clinton     sanders        cruz       obama     hillary         oil
##        1695        1206         967         702         626         618
##  photograph    violence   australia     senator     trump's    refugees
##         585         568         529         525         520         495
##    brussels      budget  australian      bernie         gun       labor
##         484         477         471         466         463         459
##      energy        food     climate         ted       talks       rubio
##         454         452         449         448         445         445
##     markets       child     primary     johnson       syria      corbyn
##         444         442         441         437         437         431
##       sales       water    officers  candidates    benefits     funding
##         423         421         417         415         412         412
##        isis       china        poll    turnbull       video       banks
##         402         401         397         396         390         388
## immigration        race       black      prices       story  supporters
##         384         382         376         375         373         367
##      senate     america    military      sector       rates    spending
##         364         362         356         355         350         343
##     cabinet   coalition   education       green  leadership    governor
##         342         340         339         336         336         335
##      french     victims development   agreement     doctors      review
##         334         333         333         332         331         330
##     project          un      kasich       rally        laws        east
##         325         324         323         322         321         318
##   emergency       fight      voting      syrian
##         318         315         314         314
```

Now we can compute the LDA with the `textmodel_lda()` function from package `seededlda`:

```
library(seededlda)
tmod_lda <- textmodel_lda(dfmat_news,
                          k = 5) #This is the number of topics that we want
```

The most significant words of each topic are obtained with `terms()`:

```
terms(tmod_lda,
      12) #For instance, the 12 words more representative
```

```
##       topic1       topic2     topic3        topic4     topic5
##  [1,] "clinton"    "violence" "australia"   "refugees" "oil"
##  [2,] "sanders"    "officers" "australian"  "brussels" "climate"
##  [3,] "cruz"       "gun"      "labor"       "syria"    "markets"
##  [4,] "obama"      "victims"  "budget"      "johnson"  "energy"
##  [5,] "hillary"    "doctors"  "turnbull"    "corbyn"   "food"
##  [6,] "trump's"    "black"    "funding"     "isis"     "water"
##  [7,] "bernie"     "military" "coalition"   "talks"    "sales"
##  [8,] "ted"        "child"    "senate"      "benefits" "china"
##  [9,] "rubio"      "video"    "spending"    "syrian"   "prices"
## [10,] "senator"    "story"    "immigration" "un"       "rates"
## [11,] "primary"    "laws"     "education"   "french"   "sector"
## [12,] "candidates" "fight"    "review"      "cabinet"  "green"
```

With this model you can calculate, for each text in the corpus, the most probable topic to which it belongs. We will use `topics()`:

```r
head(topics(tmod_lda), 20) #Only shows the 20 first but there are as many as documents
```

```
## text136751 text136585 text139163 text169133 text153451 text163885 text157885
##     topic5     topic3     topic3     topic2     topic2     topic5     topic1
## text173244 text137394 text169408 text184646 text127410 text134923 text169695
##     topic4     topic1       <NA>     topic4     topic5     topic4     topic5
## text147917 text157535 text177078 text174393 text181782 text143323
##     topic2     topic5     topic1     topic1       <NA>     topic5
## Levels: topic1 topic2 topic3 topic4 topic5
```

Notice that there are some texts that do not fit into any of the calculated topics and are marked with `<NA>`.

We can assign the topic as a new docvar with:

```r
dfmat_news$topic <- topics(tmod_lda)
```

And we can see how many texts there are for each topic with:

```r
table(dfmat_news$topic)
```

```
##
## topic1 topic2 topic3 topic4 topic5
##    270    360    330    407    493
```