Tratamiento Inteligente de Datos

Análisis de sentimientos sobre App Google Store

Matilde Cabrera González

ÍNDICE

Googleplaystore.csv

googleplaystore.csv (1.3 MB)

App	Category	Rating	Reviews	Size	Installs	Type
Application name	Category the app belongs to	Overall user rating of the app (as when scraped)	Number of user reviews for the app (as when scraped)	Size of the app (as when scraped)	Number of user downloads/installs for the app (as when scraped)	Paid or Free
9660 unique values	FAMILY 18% GAME 11% Other (32) 71%	1 – 19	0 – 78.2m	Varies with device 16% 11M 2% Other (460) 83%	1,000,000+ 15% 10,000,000+ 12% Other (20) 74%	Free 93% Paid 7% Other (2) 0%

13 of 13 columns

Views

	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
s for the ped)	Paid or Free	Price of the app (as when scraped)	Age group the app is targeted at - Children / Mature 21+ / Adult	An app can belong to multiple genres (apart from its main category). For eg, a musical family game will belong to	Date when the app was last updated on Play Store (as when scraped)	Current version of the app available on Play Store (as when scraped)	Min required Android version (as when scraped)
15% 12% 74%	Free 93% Paid 7% Other (2) 0%	0 93% $0.99 1% Other (91) 6%	Everyone 80% Teen 11% Other (5) 8%	Tools 8% Entertainment 6% Other (118) 86%	21May10 – 8Aug18	Varies with device 13% 1.0 7% Other (2832) 79%	4.1 and up 23% 4.0.3 and up 14% Other (33) 64%

Googleplaystore_user_review.csv

googleplaystore_user_reviews.csv (7.31 MB)

Views

App

Name of app

A String

1074 unique values

Valid	64.3k	100%
Mismatched	0	0%
Missing	0	0%
Unique	1074	
Most Common	Angry Bird:	0%

Translated_Review

User review (Preprocessed and translated to English)

A String

nan	42%
Good	0%
Nice	0%
Great	0%
Other (27992)	57%

Valid	64.3k	100%
Mismatched	0	0%
Missing	5	0%
Unique	28.0k	
Most Common	nan	42%

Sentiment

Positive/Negative/Neutral (Preprocessed)

A String

nan	42%
Positive	37%
Negative	13%
Neutral	8%

Valid	64.3k	100%
Mismatched	0	0%
Missing	0	0%
Unique	4	
Most Common	nan	42%

Sentiment_Polarity

Sentiment polarity score

A String

nan	42%
0.0	8%
0.5	2%
0.7	2%
Other (6489)	46%

Valid	64.3k	100%
Mismatched	0	0%
Missing	0	0%
Unique	6493	
Most Common	nan	42%

Sentiment_Subjectivity

Sentiment subjectivity score

A String

nan	42%
0.0	7%
1.0	3%
0.5	3%
Other (4691)	46%

Valid	64.3k	100%
Mismatched	0	0%
Missing	0	0%
Unique	4695	
Most Common	nan	42%

Entrenar modelos para clasificar textos según su sentimiento, analizar dependencia con otras variable y el mercado de Android.

```
App : Factor w/ 755 levels "10 Best Foods for You",..: 1 1 1 1 1 1 1 1 1 1 ...
Translated_Review : Factor w/ 4459 levels ", Bethe bethe worm will be cut, you will always h
Sentiment : Factor w/ 3 levels "Negative","Neutral",..: 3 1 3 3 2 3 3 3 3 3 ...
Sentiment_Polarity : num 0.7 -0.5 0.1 0.7 0 ...
Sentiment_Subjectivity: num 0.6 0.5 1 0.6 0 ...
Category : Factor w/ 33 levels "ART_AND_DESIGN",..: 16 16 16 16 16 16 16 16 16 16 ...
Rating : num 4 4 4 4 4 4 4 4 4 4 ...
Reviews : int 2490 2490 2490 2490 2490 2490 2490 2490 2490 2490 ...
Size : Factor w/ 163 levels "1.2M","1.3M",..: 45 45 45 45 45 45 45 45 45 45 ...
Installs : Factor w/ 12 levels "1,000,000,000+",..: 12 12 12 12 12 12 12 12 12 12 ...
Type : Factor w/ 2 levels "Free","Paid": 1 1 1 1 1 1 1 1 1 1 ...
Price : Factor w/ 8 levels "$0.99","$11.99",..: 8 8 8 8 8 8 8 8 8 8 ...
Content.Rating : Factor w/ 5 levels "Adults only 18+",..: 3 3 3 3 3 3 3 3 3 3 ...
Genres : Factor w/ 66 levels "Action","Action;Action & Adventure",..: 33 33 33 33 33 33 33 3
Last.Updated : Factor w/ 239 levels "April 1, 2016",..: 69 69 69 69 69 69 69 69 69 69 ...
Current.Ver : Factor w/ 465 levels "0.5.8","0.6.88",..: 102 102 102 102 102 102 102 102 102
Android.Ver : Factor w/ 21 levels "1.5 and up","1.6 and up",..: 7 7 7 7 7 7 7 7 7 7 ...
```

Translated_Review	Sentiment
Good	Positive
No recipe book Unable recipe book.	Negative
Wow	Positive
good food really good eat	Positive
It helpful site ! It help foods get !	Neutral
Weight loss Not bad	Positive
good you.	Positive
Great Love food	Positive
Nothing special! Could find anything useful!	Positive
Absolutely Fabulous Phenomenal	Positive
HEALTH SHOULD ALWAYS BE TOP PRIORITY. !!. ON M...	Positive
Luv it! Simple, easy understand, well thought I follo...	Positive
Great app. Love	Positive
Doesn't work... Zero	Neutral
Amazing	Positive
Best way	Positive
A big thanks ds I got bst gd health	Positive
Great wife. My wife enjoy much. She's kinda person ...	Positive
Love This really good	Positive
Good.!!	Positive
Food list easy I predibetic, I scared. All Dr. said pota...	Positive
nice super get	Positive
Faltu plz waste ur time	Negative
Very Useful in diabetes age 30. I need control sugar....	Positive
Luv	Neutral
Quick Read.	Positive
Thanks helpful app.	Positive
Great Love	Positive
Best	Positive
Crap Doesn't work	Negative
Good healthy foods.	Positive
10 best foods 4u Excellent chose foods	Positive
I do not collect it for a month, but I will not refund it...	Neutral

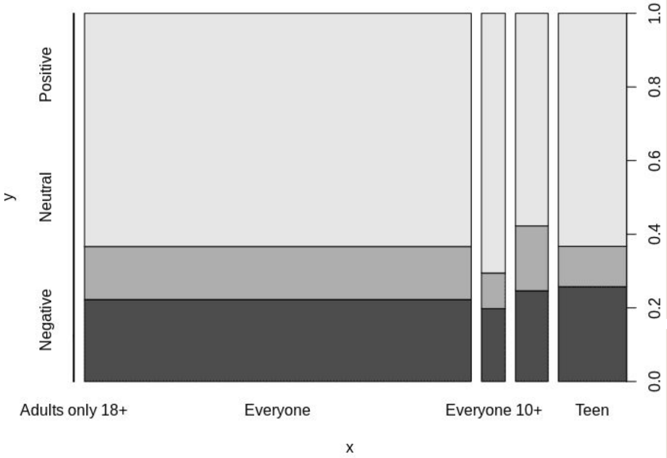
Preprocesamiento y análisis exploratorio global

```r
user_reviews <- na.omit(googleplaystore_user_reviews)
# Dividimos nuestro dataset en 3, filtrando por sentimientos:
positive <- user_reviews %>% filter(Sentiment=="Positive")
negative <- user_reviews %>% filter(Sentiment=="Negative")
neutral <- user_reviews %>% filter(Sentiment=="Neutral")
# Generamos 1500 números al azar para positive que tiene 23998 muestras.
filas.random <- sample(1:23998, 1500, replace= F)
positive <- as.data.frame(positive[filas.random,])
# Para negative:
filas.random <- sample(1:8271, 1500, replace= F)
negative <- as.data.frame(negative[filas.random,])
# Para neutral:
filas.random <- sample(1:5158, 1500, replace= F)
neutral <- as.data.frame(neutral[filas.random,])
# Ahora unimos los review a googleplaystore y quitamos los duplicados
user_reviews = rbind(positive,negative,neutral)
user_reviews <- user_reviews[!duplicated(user_reviews), ]
datospro = merge(user_reviews, googleplaystore)
# Quitamos las filas que tienen valores nulos
datospro = na.omit(datospro)
# Quitamos repetidos
datospro <- datospro[!duplicated(datospro), ]
```

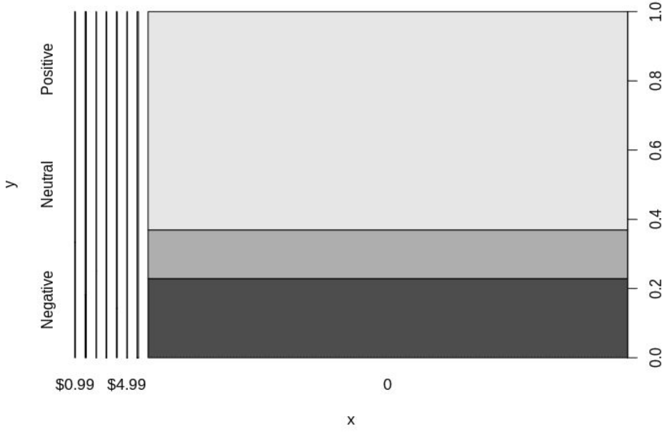
```r
Sentiment = as.factor(datospro$Sentiment)
Sentiment = factor(Sentiment, levels = c("Negative","Neutral","Positive"))
Category = as.factor(datospro$Category)
Review = as.factor(datospro$Translated_Review)
Type = as.factor(datospro$Type)
Type = factor(Type, levels = c("Free","Paid"))
Price = as.factor(datospro$Price)
Genres = as.factor(datospro$Genres)
Installs = as.factor(datospro$Installs)
Rating = as.factor(datospro$Content.Rating)
App = as.factor(datospro$App)

# visualmente

plot(Type,Sentiment, xlab="Type", ylab="Sentimientos")
plot(Review,Sentiment, xlab="Review", ylab="Sentimientos")
plot(Rating,Sentiment, xlab="Rating", ylab="Sentimientos")
plot(Price,Sentiment, xlab="Price", ylab="Sentimientos")
plot(Installs,Sentiment, xlab="Installs", ylab="Sentimientos")
plot(Genres,Sentiment, xlab="Genres", ylab="Sentimientos")
plot(Category,Sentiment, xlab="Category", ylab="Sentimientos")
plot(App,Sentiment, xlab="App", ylab="Sentimientos")
```

Type

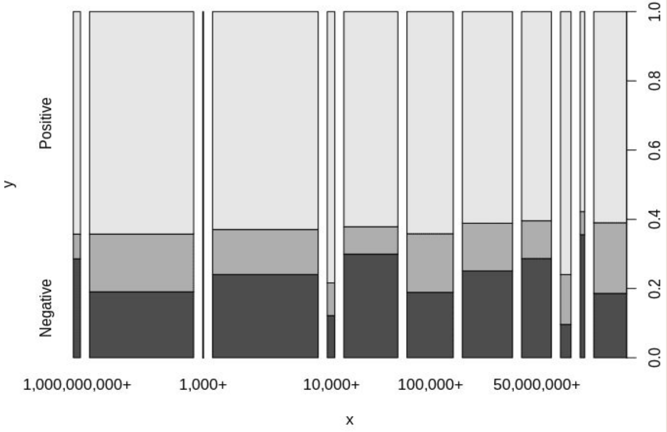
Rating



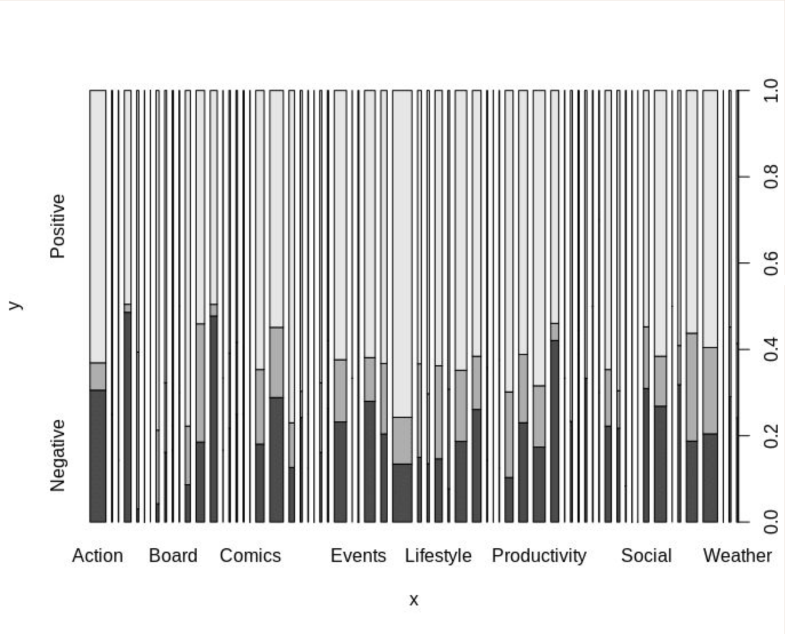
Price



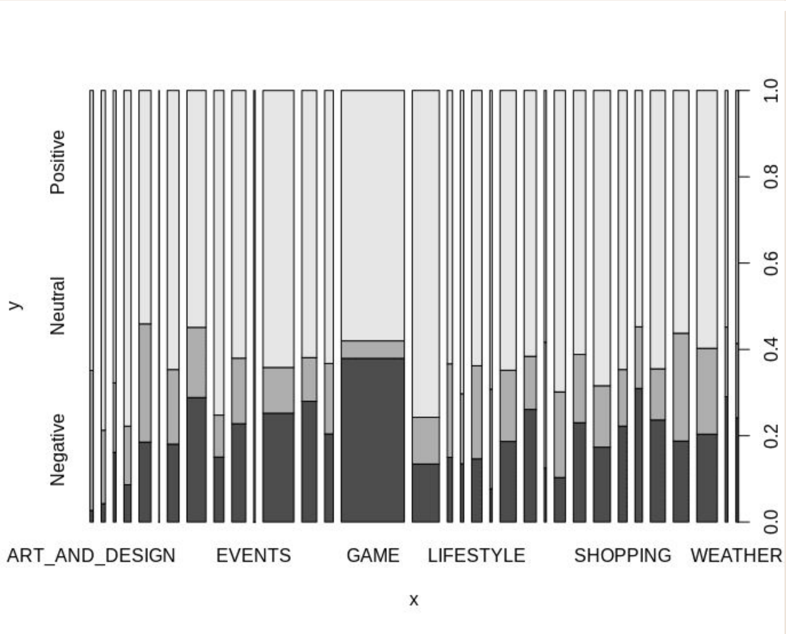
Installs



Genres



Category



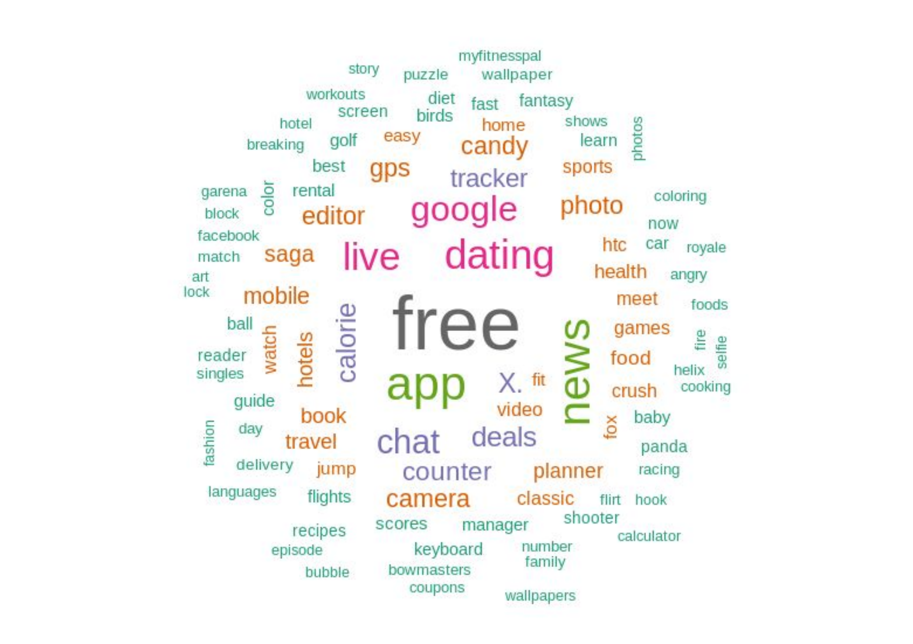
Preprocesamiento y análisis exploratorio semantico

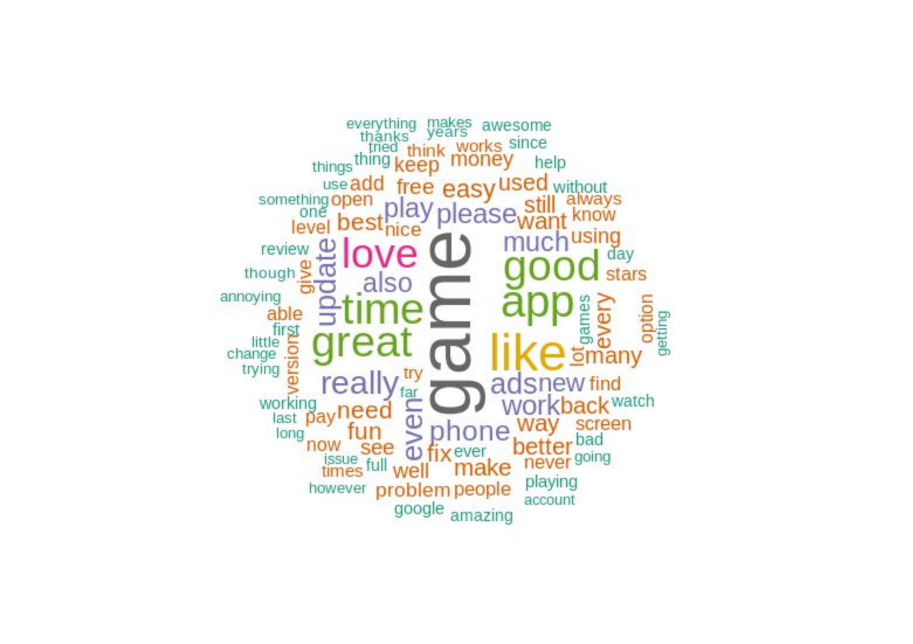
```
Trans_review = datosprocesados$Translated_Review
Trans_review = as.matrix(Trans_review)

# Elimina numeros
Trans_review <- gsub("[[:digit:]]", "", Trans_review)
# Elimina RT
Trans_review <- gsub("RT","",Trans_review)
# Elimina espacios en blanco múltiples
Trans_review <- gsub("[\\s]+", "", Trans_review)
# Elimina signos como @
Trans_review <- gsub("([^[:alpha:] ]+|&amp;|&am;|\\n+)", "", Trans_review)
# Eliminacion de signos de puntuación
Trans_review <- gsub("[[:punct:]]", "", Trans_review)
# Pasar de mayuscula a minuscula
Trans_review <- tolower(Trans_review)
```

Preprocesamiento y análisis exploratorio semántico

```
corpus.tmp <- Corpus(VectorSource(Trans_review))

#eliminar los términos con longitud menos de 3 letras y mayores de 12
dtm <- DocumentTermMatrix(corpus.tmp, list(wordLengths= c(3,10)))

#Eliminar los términos que aparecen en muy pocos documentos
inspect(dtm)
Sparsity          : 99% #umbral de escasez es 0,99

dtm <- removeSparseTerms(dtm, sparse= 0.99)
inspect(dtm
Sparsity          : 100%
```

Preprocesamiento y análisis exploratorio semántico





Agrupamiento

Preprocesamiento

```
# vector de pesos en idf
idf <- log(nrow(tf)/(colSums(tf!=0))+1)

# Matriz tf-idf
dtidf <- t(t(tf)*idf)
```

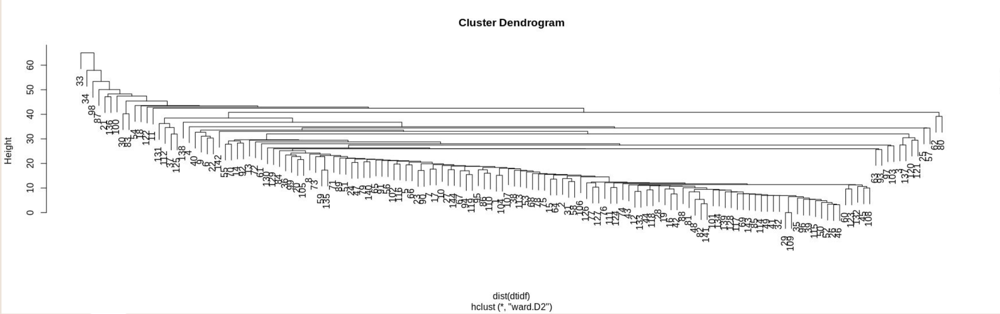
Agrupamiento Jerárquico

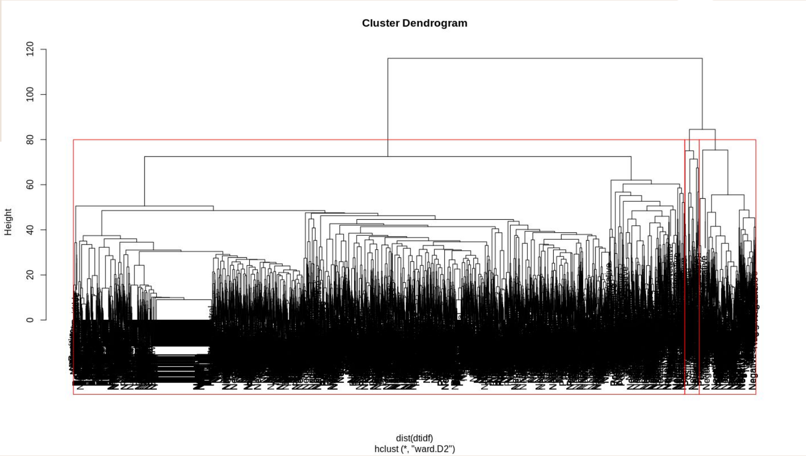
Calculamos la distancia euclídea de la matriz inversa de documento-termino

hc=hclust(dist(dtidf),method="ward.D2")

hc

Dibujamos el dendrograma y cortamos por tres

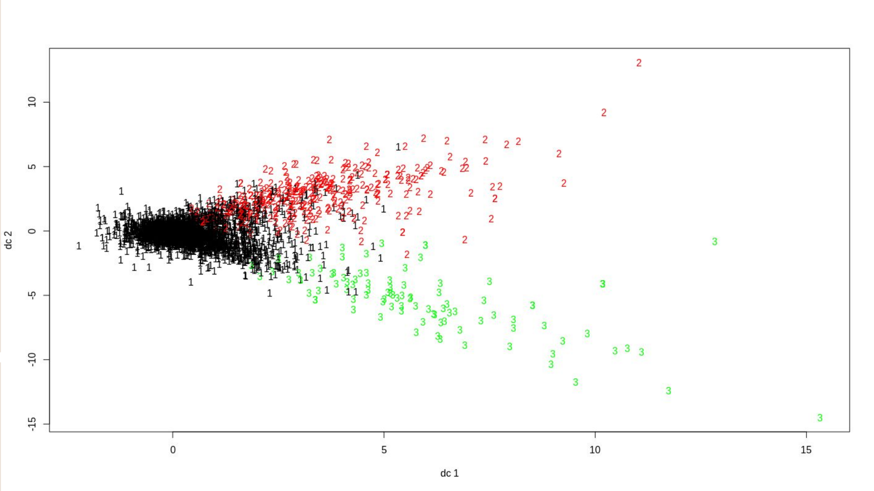


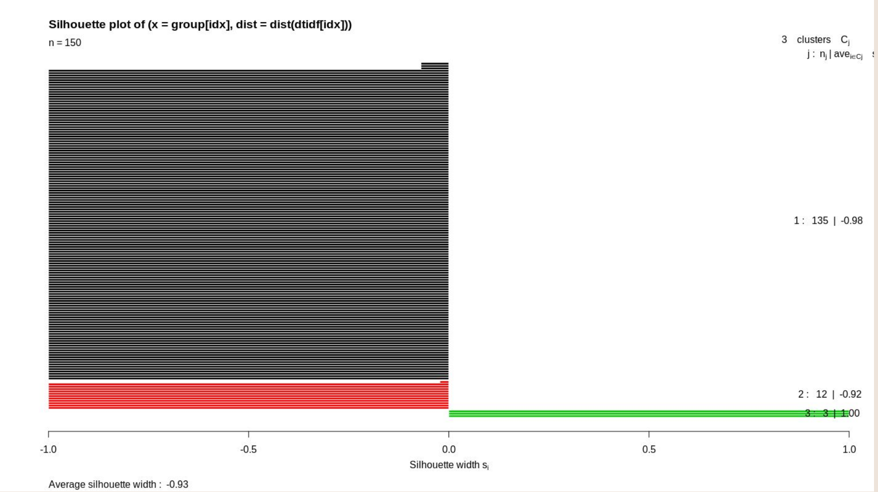


Agrupamiento Jerárquico

```
group=cutree(hc,k=3)
plotcluster(dtidf,group
```

```
idx=sample(1:dim(dtidf)[1],150)
shi= silhouette(group[idx],dist(dtidf[idx]))
plot(shi,col=1:3)
```

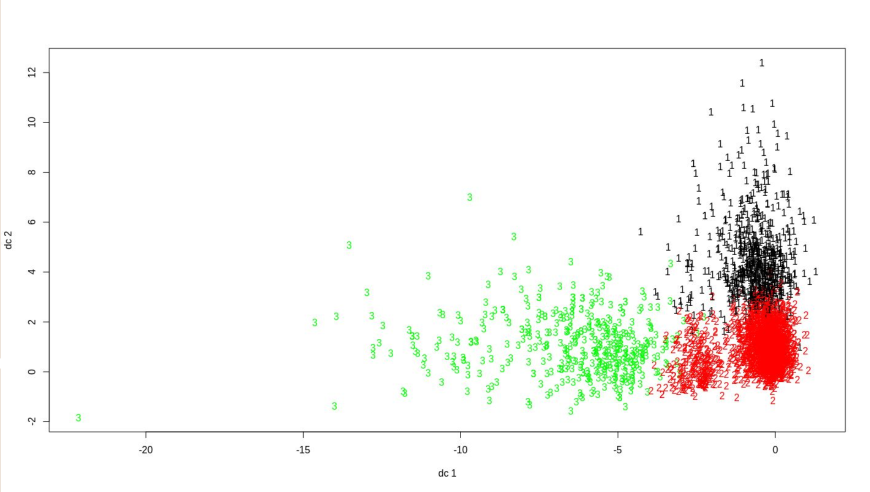


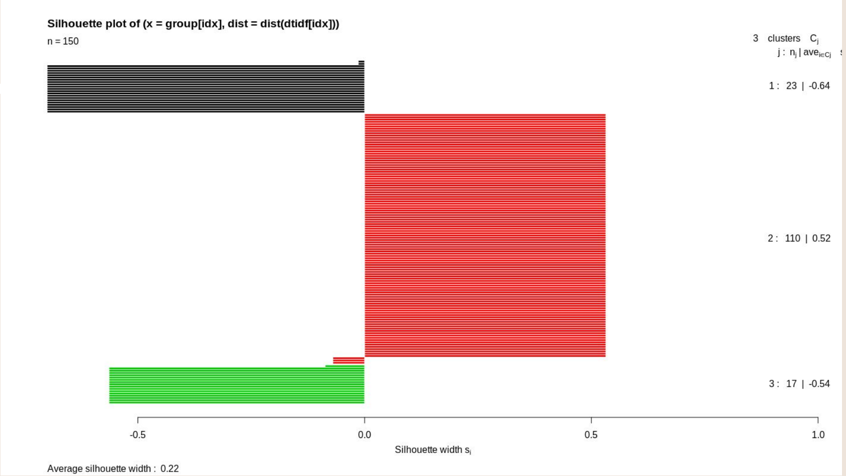


k−medias

```
kmeans.result=kmeans(dtidf,3)
kmeans.result
```

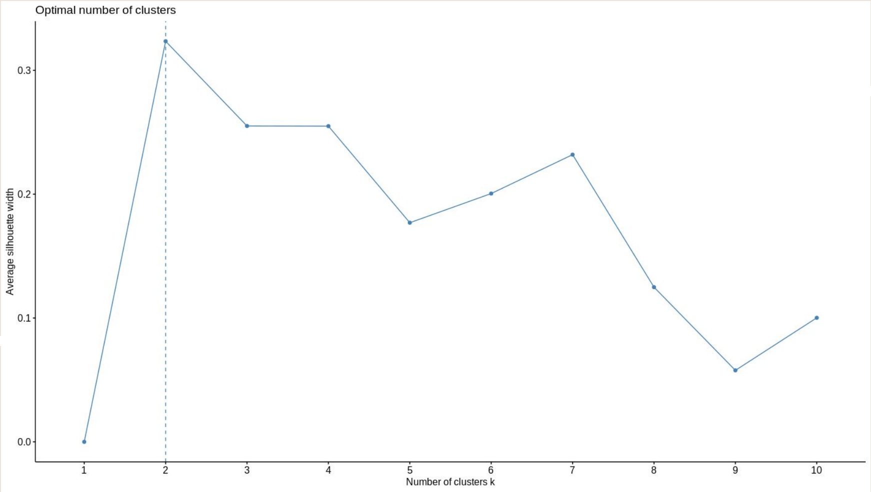
```
idx=sample(1:dim(dtidf)[1],150)
shi= silhouette(kmeans.result[idx],dist(dtidf[idx]))
plot(shi,col=1:3)
```





k-medias

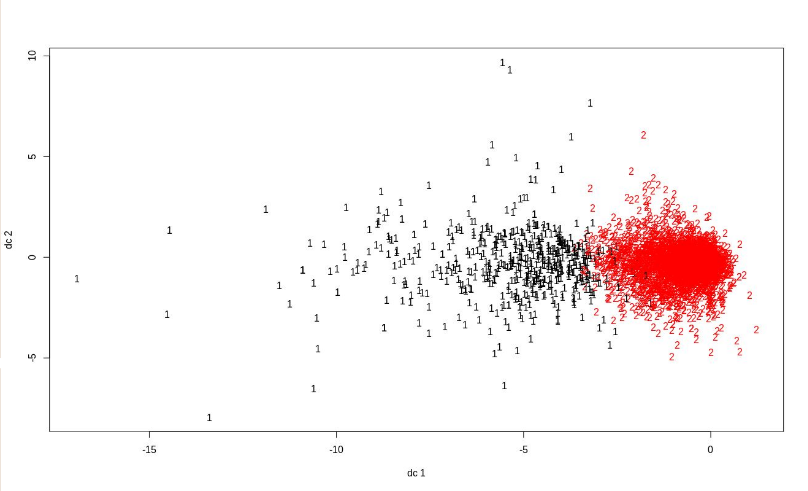
fviz_nbclust (dtidf, kmeans, method = "silhouette")

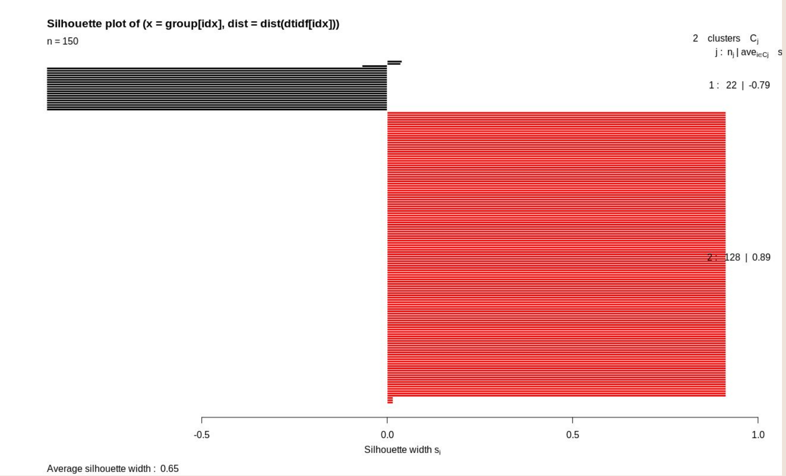


k–medias

```
kmeans.result=kmeans(dtidf,2)
kmeans.result
```

```
idx=sample(1:dim(dtidf)[1],150)
shi= silhouette(kmeans.result[idx],dist(dtidf[idx]))
plot(shi,col=1:2)
```

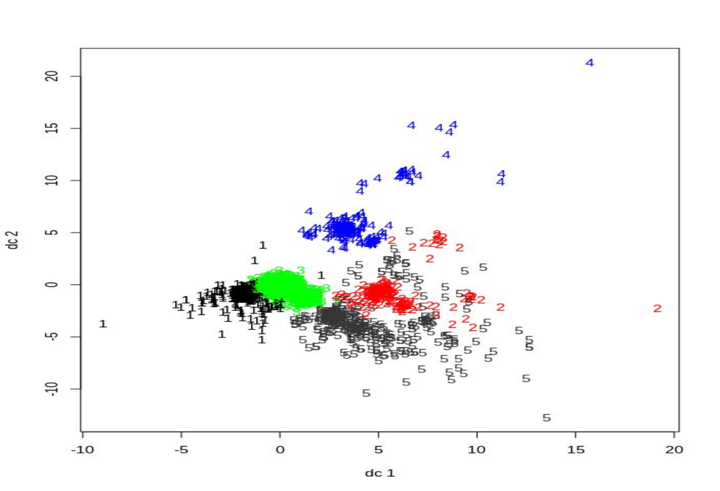


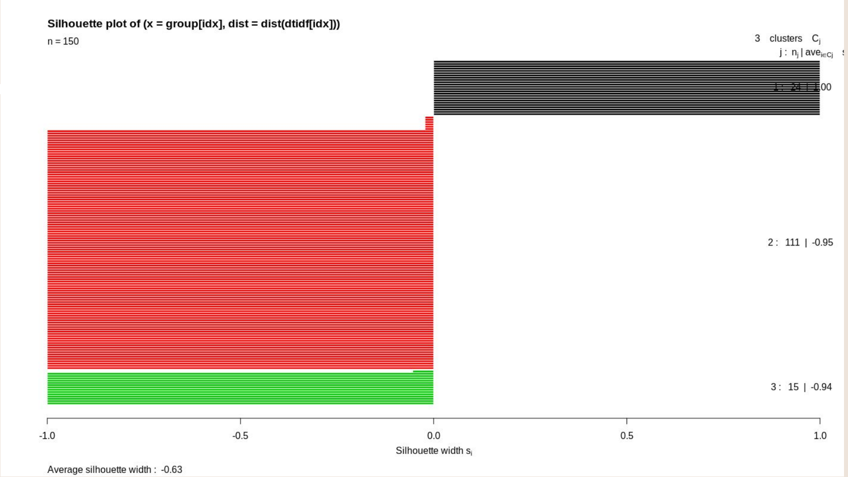


k-medoides

```
pam.result=pam(dist(dtidf),5)
idx=sample(1:dim(dtidf)[1],150)
grupo=pam.result$clustering
plotcluster(dtidf,grupo)
```

```
idx=sample(1:dim(dtidf)[1],150)
shi= silhouette(grupo[idx],dist(dtidf[idx]))
plot(shi,col=1:3)
```

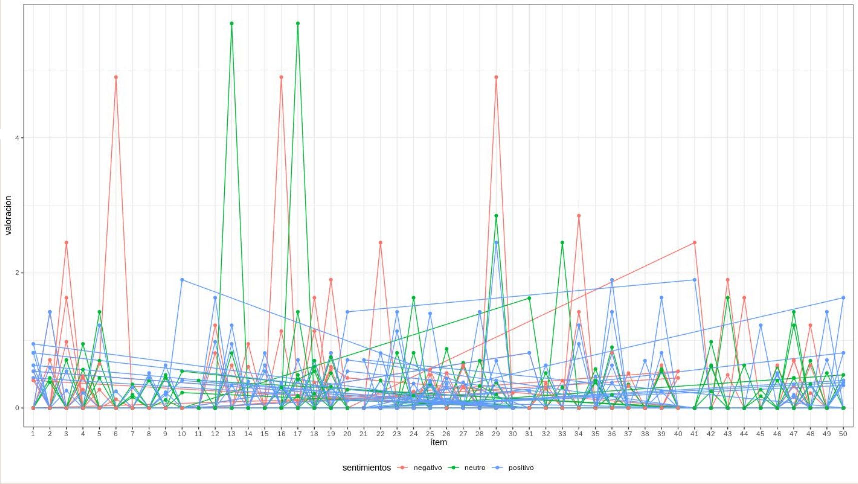




Conclusiones Agrupamiento

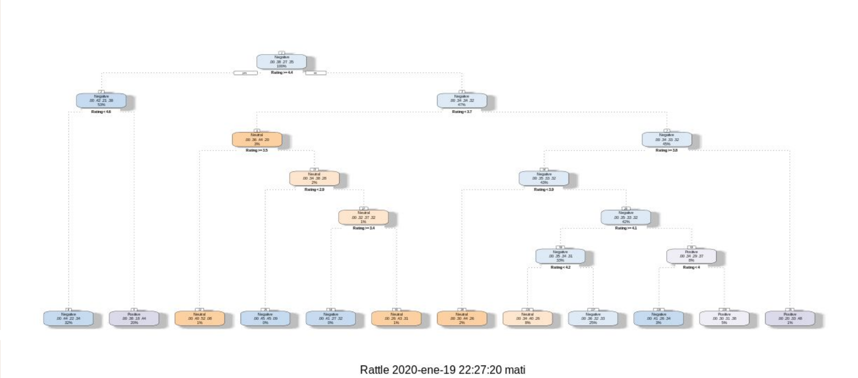
```
# Distancia coseno
tfidf_DT <- suppressWarnings(weightTfIdf(dtm))
terms_DT <- tfidf_DT$dimnames$Terms
id1 <- 1
id2 <- 2
doc1 <- as.vector(tfidf_DT[ , id1])
names(doc1) <- terms_DT
doc2 <- as.vector(tfidf_DT[ , id2])
names(doc2) <- terms_DT

distancia <- function(x, y){
  resultado <- x%*%y / (sqrt(x %*% x) * sqrt(y %*%y
))
  return(as.numeric(resultado))
}
distancia(doc1,doc2)
[1] 0.01800277
```

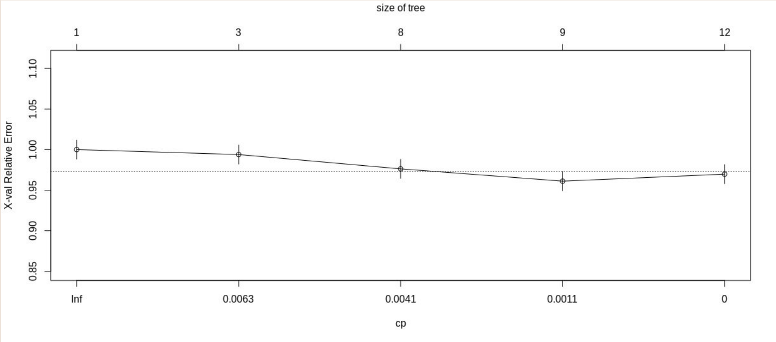
Correlación (recta lineal y proporcionalidad entre variables)



Clasificación: Árboles de decisión

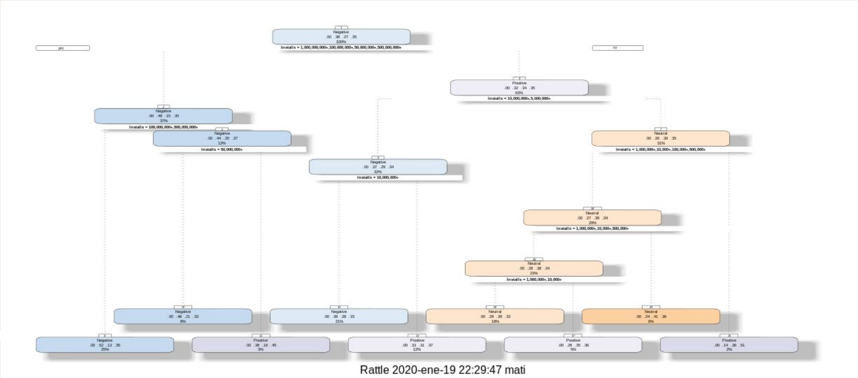
```
library("rpart")
library("rpart.plot")

c=sample(2,nrow(Trans_review),replace=TRUE,prob
=c(0.7,0.3))
user_train <- Trans_review[c==1,]
user_test <- Trans_review[c==2,]

rpart <- rpart(Sentiment ~ Rating, data=user_train,
            method="class",
            parms=list(split="information"),
            control=rpart.control(minsplit=30,
                        minbucket=10,
                        cp=0.01,
                        usesurrogate=0,
                        maxsurrogate=0)
)
fancyRpartPlot(rpart)
```

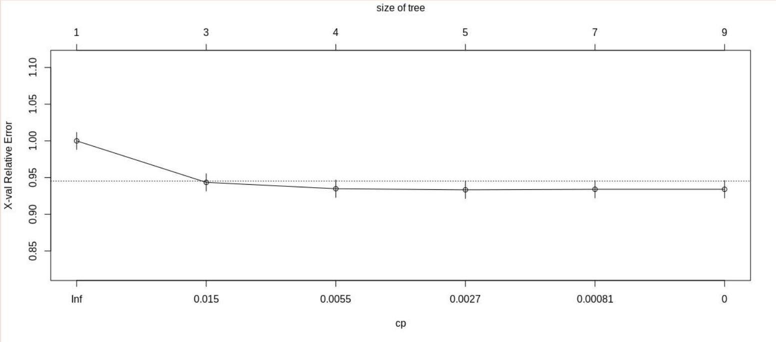




Clasificación: Árboles de decisión

```
rpart <- rpart(Sentiment ~ Installs, data=user_train,
          method="class",
          parms=list(split="information"),
          control=rpart.control(minsplit=30,
                         minbucket=10,
                         cp=0.00,
                         usesurrogate=0,
                         maxsurrogate=0)
)
fancyRpartPlot(rpart)
```

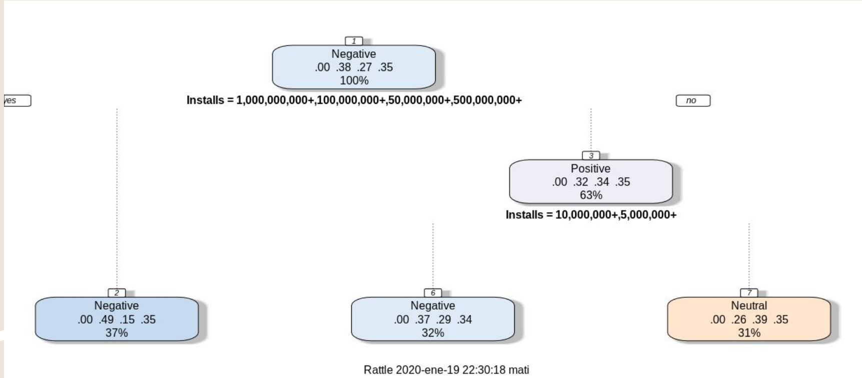




Clasificación: Árboles de decisión

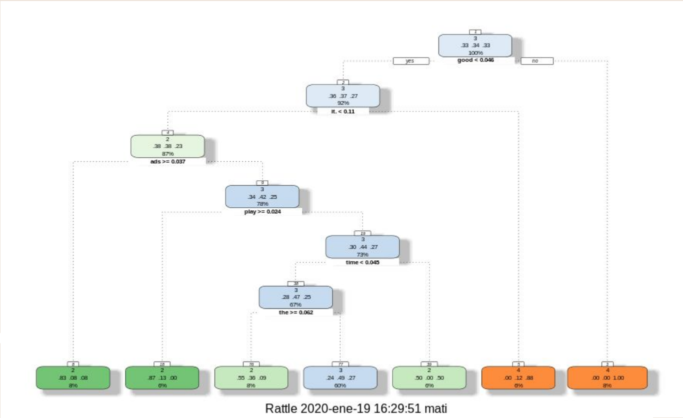
cp = 0.015

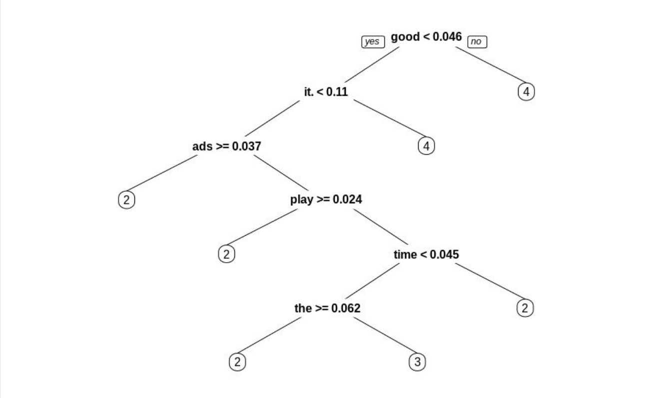
```
rpart <- rpart(Sentiment ~ Installs, data=user_train,
     method="class",
     parms=list(split="information"),
     control=rpart.control(minsplit=30,
     minbucket=10,
     cp=0.015,
     usesurrogate=0,
     maxsurrogate=0))
fancyRpartPlot(rpart)
```



Clasificación: Árboles de decisión

```
datos_total <- bind_rows(user_train,user_test)
corpus =
Corpus(VectorSource(datos_total$Translated_Review))
tdm <- tm::DocumentTermMatrix(corpus)
tdm.tfidf <- tm::weightTfIdf(tdm)
reviews = as.data.frame(cbind(datos_total$Sentiment,
as.matrix(tdm.tfidf)))
reviews <- na.omit(reviews)
preparado_train <- reviews[1:1500,]
preparado_test <- reviews[-(1:1500),]
reviews_tree = rpart(V1~., method = "class", data=
preparado_train)
prp(reviews_tree)
```





Clasificación: SVM

```
corpus = Corpus(VectorSource(user_train$Translated_Review))
tdm <- DocumentTermMatrix(corpus)
#tdm.tfidf <- suppressWarnings(weightTfIdf(tdm))

container <- create_container(tdm, t(user_train$Sentiment),
                        trainSize = 1:2996,
                        virgin = FALSE)

models <- train_models(container,algorithms=c("SVM"))

results <- classify_models(container, models)

out = data.frame(model_sentiment = results$SVM_LABEL,
          model_prob = results$SVM_PROB,
          actual_party = user_train$label[1:2999])
(z = as.matrix(table(out[,1], out[,3])))   # display the confusion matrix.
```

	Negative	Neutral	Positive
Negative	833	18	34
Neutral	91	957	78
Positive	40	15	933

La precisión (accuracy))

```
[1] 90.73
```

Clasificación: SVM

```
text = user_train$Translated_Review
cor = Corpus(VectorSource(text)) # crea otra vez el corpus
dtm <- DocumentTermMatrix(cor, list(bounds= list(global= c(5,Inf))))
dtm.test <- suppressWarnings(weightTfIdf(dtm))
row.names(dtm.test) = (nrow(dtm)+1):(nrow(dtm)+nrow(dtm.test))
dtm.f = c(tdm, dtm.test)
training_codes.f = c(training_codes,
                     rep(NA, length(user_train)))

container.f = create_container(dtm.f,
                     t(training_codes.f), trainSize=1:nrow(tdm),
                     testSize = 1:1291, virgin = T)
model.f = train_models(container.f, algorithms = c("SVM"))
predicted <- classify_models(container.f, model.f)
out = data.frame(model_sentiment = predicted$SVM_LABEL,
               model_prob = predicted$SVM_PROB,
               text = user_test$label)
(z = as.matrix(table(out[,1], out[,3])))
```

	Negative	Neutral	Positive
1	14	211	18
2	32	54	25
3	396	161	380

La precisión (accuracy))

```
[1] 34.7
```

Clasificación: KNN

```
c=sample(2,nrow(user_reviews),replace=TRUE,prob=c(0.7,0.3))
data_train <- user_reviews[c==1,]
data_test <- user_reviews[c==2,]

colnames(data_train)<-c("text", "label")
colnames(data_test)<-c("text", "label")

# si los datos no son numericos introduce NAS por defecto
data_train$text <- as.numeric(data_train$text)
data_test$text <- as.numeric(data_test$text)
trainClass<-data_train[,"label"]
trueClass<-data_test[,"label"]
knnClass <- knn (data_train, data_test, trainClass)
# Matriz de confusión:
nnTabla <- table ("1-NN" = knnClass, Reuters = trueClass); nnTabla
```

1-NN	Negative	Neutral	Positive
Negative	190	102	102
Neutral	131	221	102
Positive	107	97	253

La precisión

```
sum(diag(nnTabla))/nrow(data_test)
[1] 0.5243446
```

Clasificación: Naive Bayes

```
c=sample(2,nrow(user_reviews),replace=TRUE,prob=c(0.7,0.3))
user_train <- user_reviews[c==1,]
user_test <- user_reviews[c==2,]

corpus_train <- Corpus(VectorSource(user_train))
corpus_test <- Corpus(VectorSource(user_test))
dtm_train <- DocumentTermMatrix(corpus_train, list(wordLengths= c(3,12)))
dtm_test <- DocumentTermMatrix(corpus_test, list(wordLengths= c(3,12)))
```

Obtenemos la matriz de términos en valores binarios en lugar de pesos

```
train_DT <- apply(dtm_train, MARGIN = 2, convert_binary)
test_DT <- apply(dtm_test, MARGIN = 2, convert_binary)
```

Función usada para convertir en valor binario:

```
convert_binary <- function(x) {
  x <- ifelse(x > 0, "Yes", "No")
}
```

Clasificación: Naive Bayes

Entrenamos el clasificador con el conjunto de entrenamiento:

```
classifier <- naiveBayes(train_DT, as.factor(user_reviews$Sentiment), laplace = 1)
```

Generamos las predicciones sobre el conjunto de entrenamiento y tes

```
pred_train <- predict(classifier, newdata=train_DT)
pred_test <- predict(classifier, newdata=test_DT)
table(pred=pred_train,real=user_reviews$Sentiment)
```

pred	Negative	Neutral	Positive
Negative	728	8	14
Neutral	309	1067	336
Positive	21	4	716

```
table(pred=pred_test,real=user_reviews$Sentiment)
```

pred	Negative	Neutral	Positive
Negative	258	25	61
Neutral	120	382	140
Positive	64	14	233

error_train

```
21.6047455510459  %
```

error_test

```
32.6908249807247  %
```

Clasificación: Naive Bayes

10 palabras más usadas en los texto

```
freq.words <- findFreqTerms(dtm_train, 10)
```

Generamos las predicciones sobre el conjunto de entrenamiento y tes

```
dtm_freq_train <- DocumentTermMatrix(corpus_train, control=list(dictionary = freq.words))
dtm_freq_test <- DocumentTermMatrix(corpus_test, control=list(dictionary = freq.words))

 train_DT <- apply(dtm_train, MARGIN = 2, convert_binary)
 test_DT <- apply(dtm_test, MARGIN = 2, convert_binary)
```

table(pred=pred_train)

pred	Negative	Neutral	Positive
Negative	678	45	73
Neutral	268	981	301
Positive	79	19	672

table(pred=pred_test)

pred	Negative	Neutral	Positive
Negative	268	36	70
Neutral	135	410	136
Positive	72	9	248

error_train

```
25.1925545571245  %
```

error_test

```
33.092485549133  %
```

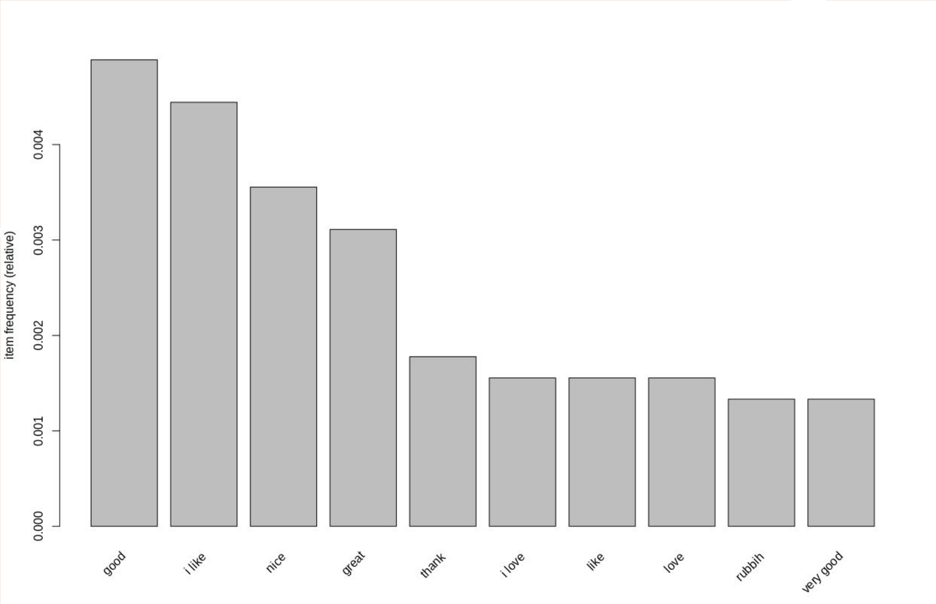
Asociación

dataset = read.csv('/home/mati/Trans_review.csv', header = FALSE)

dataset = read.transactions('/home/mati/Trans_review.csv', sep = ',', rm.duplicates = TRUE)

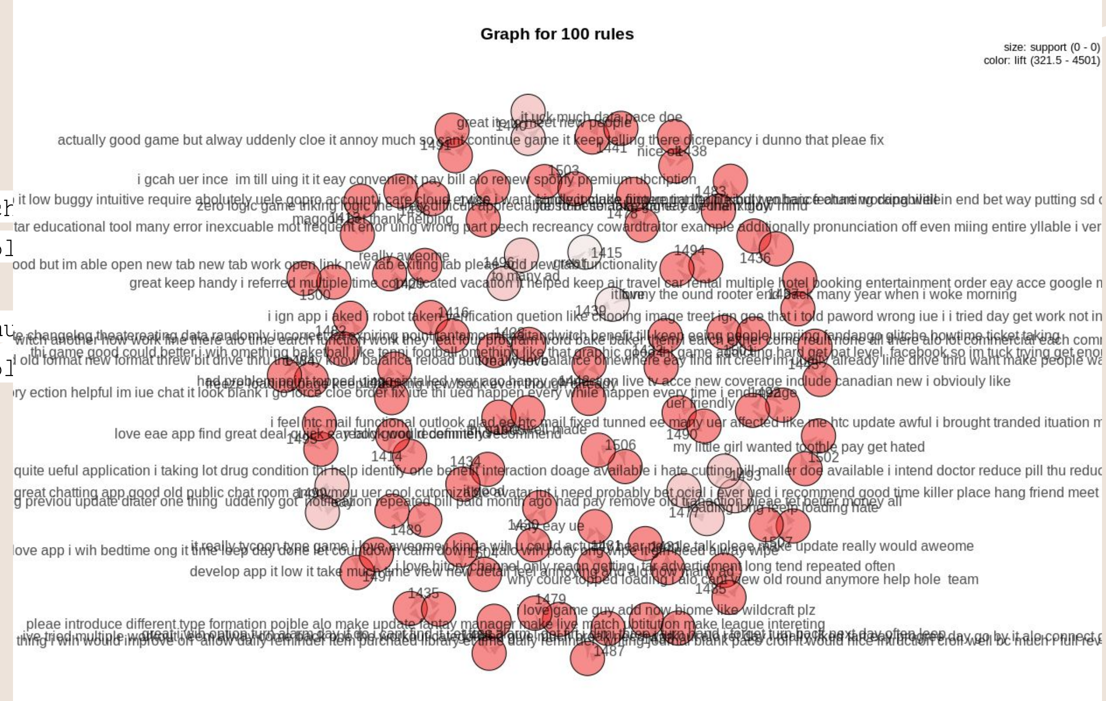
```
transactions as itemMatrix in sparse format with
 4501 rows (elements/itemsets/transactions) and
 8655 columns (items) and a density of 0.0002309006
most frequent items:
   good     like     nice    great    thank  (Other)
     22       20       16       14        8     8915
element (itemset/transaction) length distribution:
sizes
   1    2
   7 4494
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   2.000   1.998   2.000   2.000
```

itemFrequencyPlot(dataset, topN = 10)



Asociación

```
rules = apriori(data = dataset, parameter = list(support = 0.00001, confidence = 0.2))
inspect(sort(rules, by = 'lift')[1:10])
     lhs
rhs
support confidence lift count
[1]  {2958}
=> {i like except ad make impoible get much
photo i took that wih ad alway cauing probl
0.0002221729          1 4501      1
[2]  {i like except ad make impoible get mu
photo i took that wih ad alway cauing probl
=> {2958}
0.0002221729          1 4501      1
```




```
plot(rules, method="graph")
```

Bibliografía

https://rpubs.com/

https://www.rdocumentation.org/

http://eio.usc.es/

https://www.kaggle.com/