# Trabajo Final
# Extracción, Análisis y Clasificación de una Web con reseñas de películas

Facultad De Ingeniería, Universidad De Cuenca
TEXT MINING
Freddy L. Abad L.
{freddy.abadl}@ucuenca.edu.ec

## REQUISITO 1

Recopilación de datos no estructurados desde alguna página Web con las revisiones y la etiqueta sobre la valoración de la revisión ya sean positivas y negativas. Los datos deberán ser grabados en algún repositorio (por ejemplo una hoja de cálculo), en el cual consten dos columnas. La primera columna es el contenido de la revisión y la segunda columna es la etiqueta de valoración. Para obtener la etiqueta de valoración se puede usar el ranking basado en estrellas el cual es un elemento de calificación muy común en distintas páginas web.

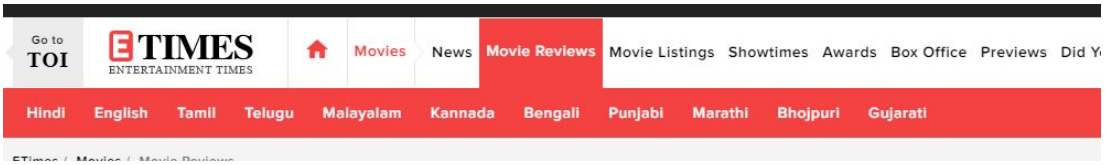
Resultados:

Almacenar un conjunto de datos similar para las reseñas positivas o negativas

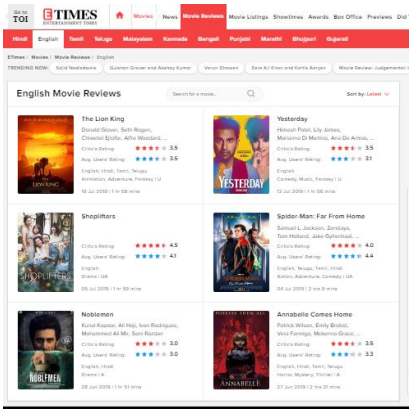
Dividir el conjunto de datos de manera aleatoria en dos mitades: el primer 50% de los datos se usará como datos de entrenamiento y el 50% restante se reserva para verificar el rendimiento del algoritmo entrenado.

**DESARROLLO - REQUISITO 1**

La pagina a analizar es: https://timesofindia.indiatimes.com/entertainment/movie-reviews esta pagina contiene reseñas de peliculas en diversos idiomas:




Debido al alcance que nos da la herramienta RapidMiner, se elige las reseñas en inglés. Estas reseñas están en un  el cual contiene a cada una de las tarjetas con las películas.

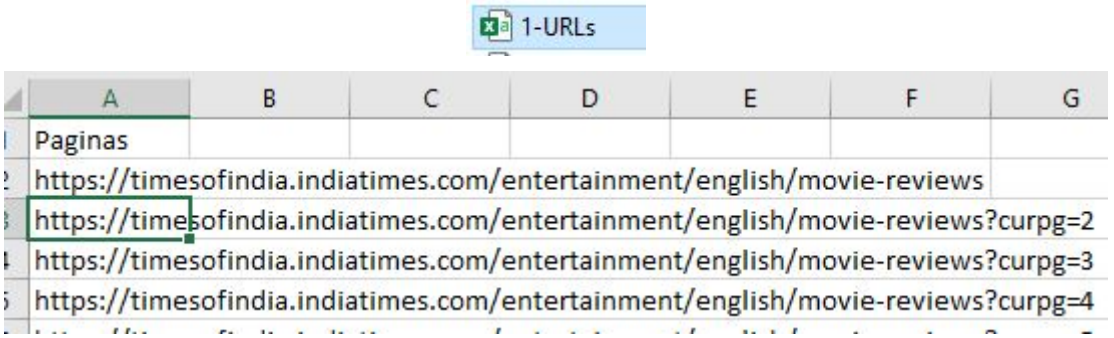



La estructura de cada una de las urls, las cuales contienen a estas reseñas se dan de la siguiente forma:

```
https://timesofindia.indiatimes.com/entertainment/english/movie-reviews
```

| |
|---|
| https://timesofindia.indiatimes.com/entertainment/english/movie-reviews?curpg=2 |
| ... |
| https://timesofindia.indiatimes.com/entertainment/english/movie-reviews?curpg=27 |

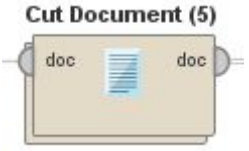
Estas URLs se almacenan en un archivo csv con nombre único, este archivo debe tener una cabecera de nombre “Páginas”:




La obtención de las urls de cada una de las urls que contiene la información de cada película se realiza en el proceso : **1-1-ConseguirUrlsPaginas.rmp**


Este proceso lee las múltiples páginas con información indexada a cada una de las películas, estas páginas contienen 50 películas por página, en un total de 27 páginas,obteniendo un total estimado de 1280 películas(la página 27).
Cada uno de las páginas se procesan con un operador “**Process Documents from Data**”, este operador nos da un vector TF-IDF.

Al interior de este operador se debe colocar un operador Cut Document, el cual extrae la información de la Figura 2, este operador extrae esta información en un atributo de nombre **PaginaPeliculas,** esta contiene cada tarjeta de cada película la cual extraemos los urls que dirigen a las reseñas de películas.

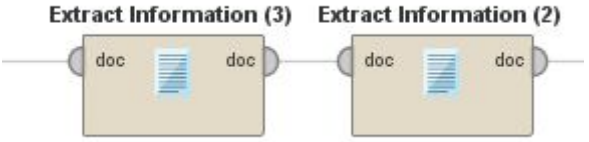



| attribute name | query expression | |
|---|---|---|
| PaginaPeliculas | <div class="ent_middle clearfi | <footer xmlns:g= |

Al interior de este operador, se coloca al igual otro operador **Cut Document,** este operador extrae el  que contiene el id de cada película

| attribute name | query expression | |
|---|---|---|
| Pelicula |  | "> </a> |

Al interior de este operador se colocan 2 operadores **Extract Information** , estos operadores extraerán el ID y URL de cada película.



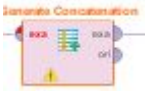

| attribute name | query expression | |
|---|---|---|
| URL1 | " href="/ | "><|

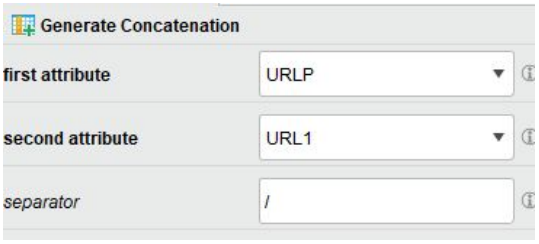
| attribute name | query expression | |
|---|---|---|
| ID | /movie-review/ | .cms |

Dado que el atributo URL1, produce una url que necesita de un complemento para generar el url exacto para el ingreso a cada una de las peliculas, se utiliza el operador **Generate Atribute**

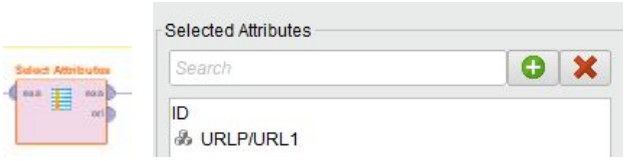
| attribute name | function expressions |
|---|---|
| URLP | "https://timesofindia.indiatimes.com" |

Posterior a este paso se concatena el Atributo IDP y ID1 mediante el operador Generate Concatenation

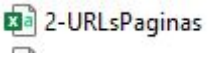


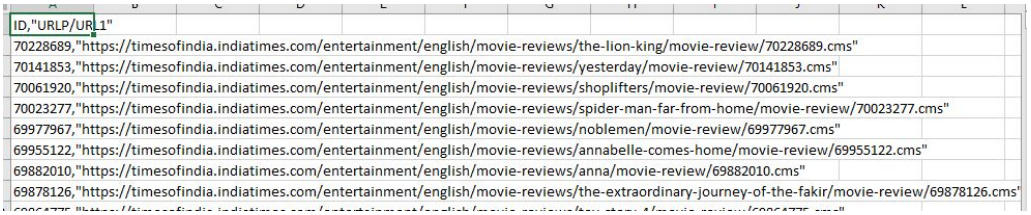



Dado el url generado, seleccionamos mediante el operador **Select Atribute,** el atributo recientemente generado y su respectivo ID



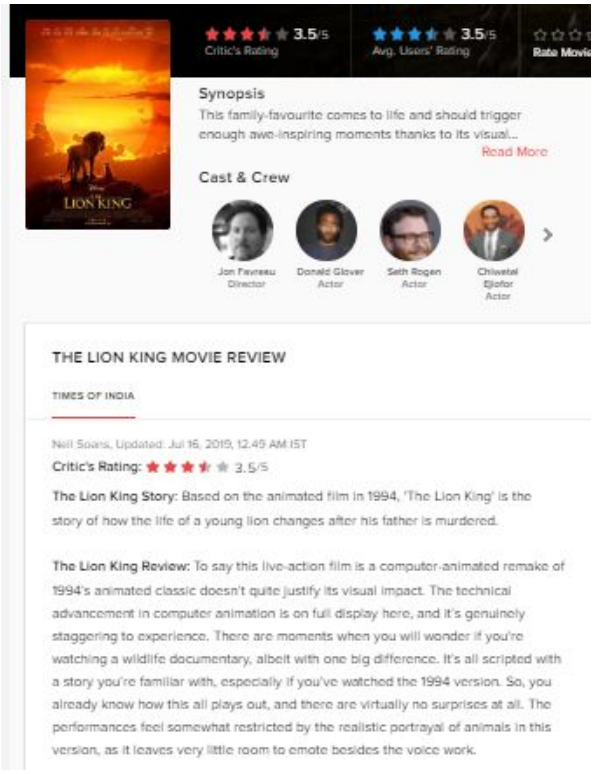

Esta nueva información se guarda en un archivo csv, el cual tendrá el nombre **2-URLsPaginas.csv,** el cual tendrá cabeceras ID, "URLP/URL1"



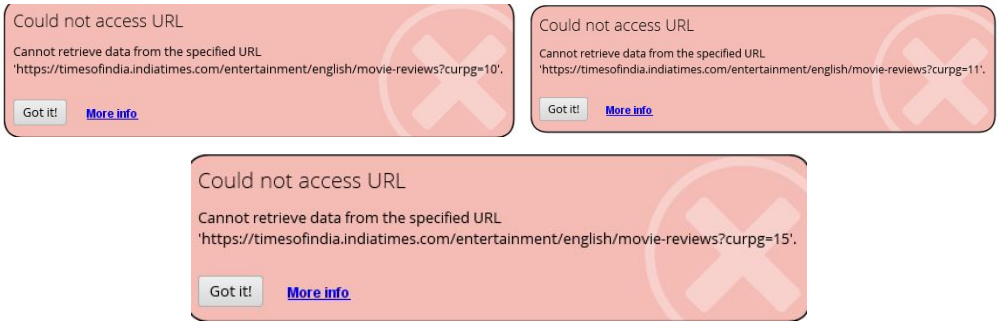


| ID | URLP/URL1 |
|---|---|
| 70228689 | https://timesofindia.indiatimes.com/entertainment/english/movie-reviews/the-lion-king/movie-review/70228689.cms |
| 70141853 | https://timesofindia.indiatimes.com/entertainment/english/movie-reviews/yesterday/movie-review/70141853.cms |
| 70061920 | https://timesofindia.indiatimes.com/entertainment/english/movie-reviews/shoplifters/movie-review/70061920.cms |
| 70023277 | https://timesofindia.indiatimes.com/entertainment/english/movie-reviews/spider-man-far-from-home/movie-review/70023277.cms |
| 69977967 | https://timesofindia.indiatimes.com/entertainment/english/movie-reviews/noblemen/movie-review/69977967.cms |
| 69955122 | https://timesofindia.indiatimes.com/entertainment/english/movie-reviews/annabelle-comes-home/movie-review/69955122.cms |
| 69882010 | https://timesofindia.indiatimes.com/entertainment/english/movie-reviews/anna/movie-review/69882010.cms |
| 69878126 | https://timesofindia.indiatimes.com/entertainment/english/movie-reviews/the-extraordinary-journey-of-the-fakir/movie-review/69878126.cms |

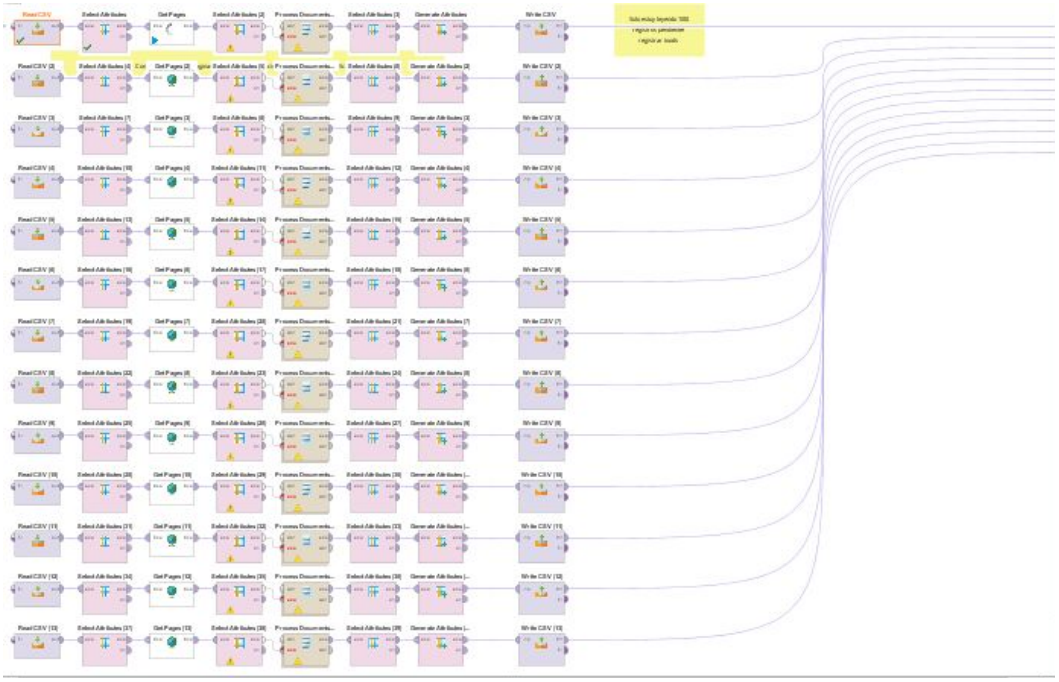
Dadas estas urls, se realiza otro proceso el cual extraiga únicamente la reseña y la calificación. Esta página tiene la siguiente estructura, la cual, en código HTML se debe procesar para obtener los atributos realmente necesarios (Reseña y Calificación), y eliminar cualquier otro atributo innecesario.



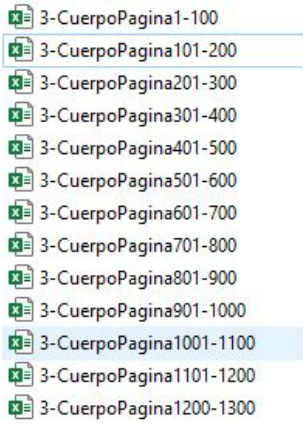

Sin embargo, al acceder a las más de 1280 páginas, el servidor de esta página considera que nuestras peticiones son muestra de un Ataque de Denegación de Servicios, dicho esto nos muestra un error, este error se "soluciona" volviendo a correr el proceso.




Este error se muestra cada 3 accesos a las páginas, dado que el número de solucion de este error es alto, y toma mucho tiempo, además de no guardar los datos de cada página accedida. Una solución que se plantea es separar el número de peticiones a 100 cada uno, obteniendo 13 subprocesos para obtener los datos necesarios. Este proceso se llama **1-2-ConseguirCorpusDividido.rmp**

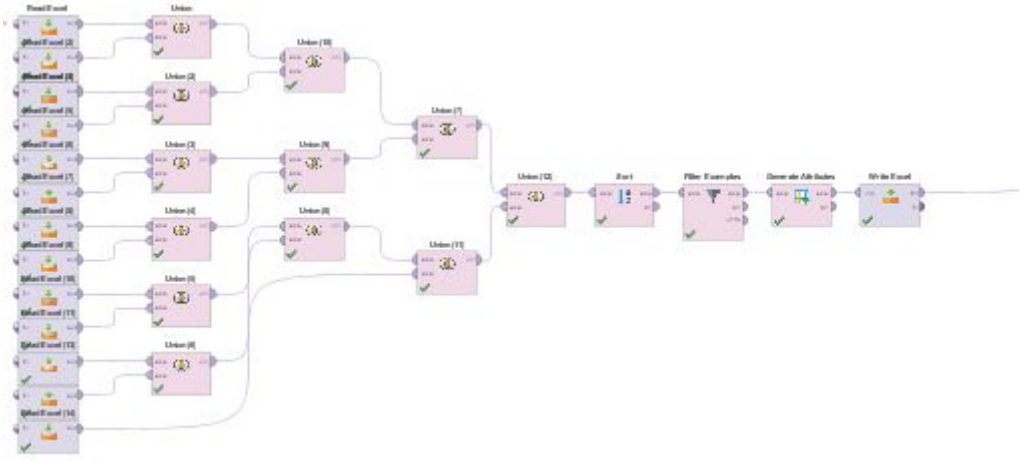


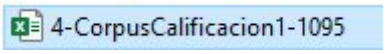
Dado todos estos procesos, (el más demorado de todo el trabajo), se obtienen 13 archivos con el siguiente formato




| | A | B | C |
|---|---|---|---|
| 1 | Cuerpo | Calificacion | |
| 2 | <div class= | 2.5 | |
| 3 | <div class= | 3.0 | |
| 4 | <div class= | 3.0 | |
| 5 | <div class= | 4.0 | |
| 6 | <div class= | 2 | |
| 7 | <div class= | 3.0 | |
| 8 | <div class= | 4 | |
| 9 | <div class= | 4 | |
| 10 | <div class= | 3.5 | |
| 11 | <div class= | 4.0 | |

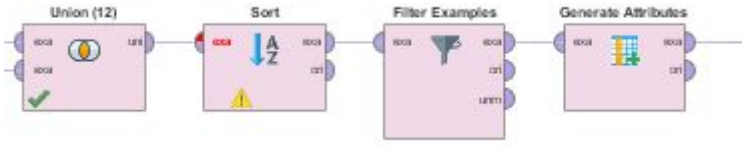
Obtenido estos archivos se procede a unirlos en un solo archivo con el proceso **1-3-UnionCorpus.rmp** con los operadores **Union**






Obtenido la unión de estos archivos, se procede a ordenar incrementalmente, sin embargo, algunos casos de estudio presentan anomalías, tales como no presentar una calificación. En estas circunstancias, el proceso obtiene un datos "**missing**" o su equivalente "**?**".

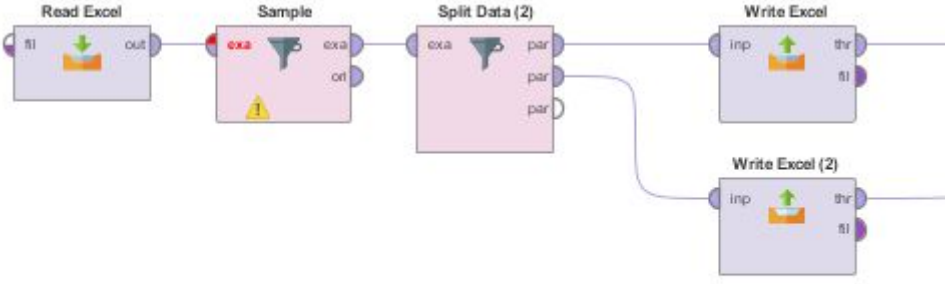

Debido a esto, se deben filtrar los casos donde no se tienen una calificación, mediante el operador **Filter Examples,** donde la Calificación tenga valores distintos de **?**





En total, se eliminan 181 registros sin calificación. A continuación, se procede a etiquetar cada película, mediante el operador **Generate Attribute.** Este operador se utiliza mediante un if-else, si su valor es menor a 3, el comentario será negativo, caso contrario será Positivo.


A continuacion, se procede a dividir en datos de prueba y datos de verificación, mediante los operadores **Sample** y **Split Data**, en el proceso llamado **1-4-SeparacionDatosTrainTest.rmp**




La configuración de lectura del archivo de entrada es la siguiente, debido a que los siguientes operadores necesitan manejar atributos de tipo **label.**

| column index | attribute meta data information | | | |
|---|---|---|---|---|
| 0 | Cuerpo | column ... | polynomi... | attribute |
| 1 | Calificacion | column ... | real | attribute |
| 2 | Etiqueta | column ... | polynomi... | label |

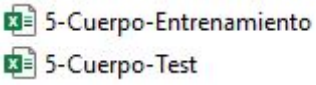
Las configuraciones de Sample es 495 en el caso de Positivo y Negativo, debido a una separación equilibrada.

| class | size |
|---|---|
| Positivo | 495 |
| Negativo | 495 |

La configuración de Split Data se configura de la siguiente manera

| ratio |
|---|
| 0.5 |
| 0.5 |

El resultado de esto produce los archivos

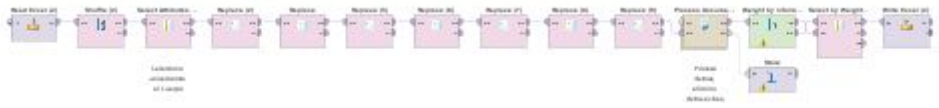



Cada uno con 247 datos positivos y negativos cada uno.

**Requisito 2:**

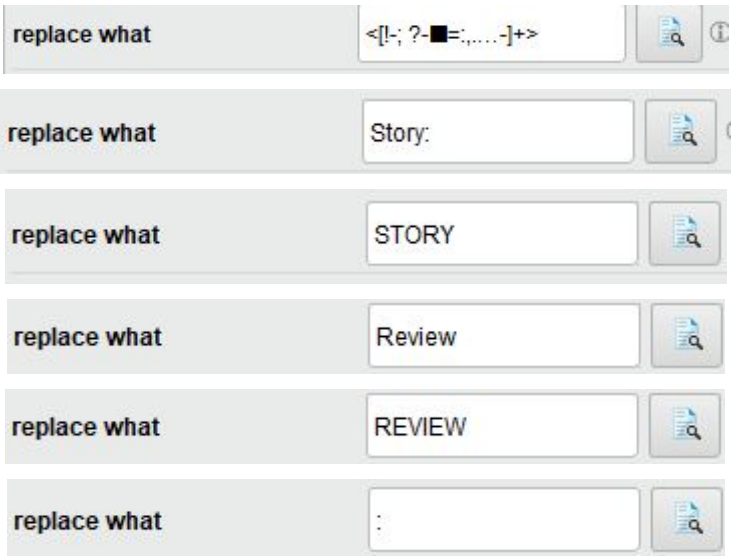
Preparación de los Datos. En este proceso la idea es convertir los datos no estructurados en un formato estructurado. Para ello ejecute todas las actividades de preprocesamiento que considere conveniente. Recuerde que si esta tarea se ejecuta de forma equivocada puede afectar seriamente la precisión del modelo entrenado. El único punto a tener en cuenta aquí es que necesitará algún operador que permita eliminar cualquier "nbsp" (se usa "& nbsp" para representar un espacio de no separación en las páginas webs) y por supuesto los comandos propios de HTML. Por tanto una actividad importante a incluir durante este proceso es i) eliminar estos caracteres del texto especificado dentro de cada fuente de datos. La salida de este proceso es un vector de documentos que especifica las palabras que constan en el blog junto con sus pesos.

Archivo **2-ProcesamientoDatosTrain.rmp**

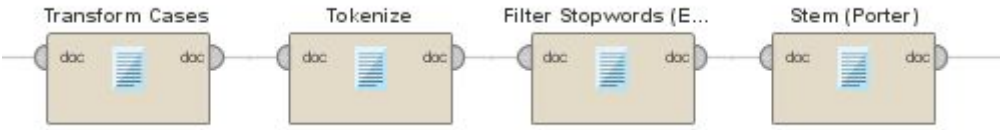


Convertir los datos no estructurados en un formato estructurado, el proceso que desarrolla este punto se llama **2-ProcesamientoDatosTrain.rmp,** la entrada es el archivo **5-CuerpoEntrenamiento.xlsx.**
La configuración de **Replace** tiene la siguiente forma

| replace what |   |
|---|---|




Y la configuración del Process Document from Data es el siguiente

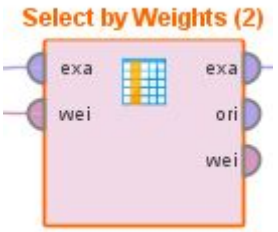


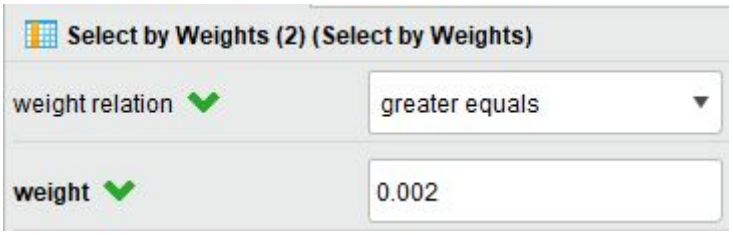

Es importante guardar el diccionario de palabras que permitirá el entrenamiento del modelo, en los pasos posteriores. Se realiza mediante el proceso **Store**

En cuanto al operador **Weight by Information Gain**, el cual es un método de selección de características, debido al gran número de palabras, se filtran por su relevancia. Se usó el operador Weight of Information Gain en lugar de Weight by SVM debido a la documentación existente, el primer operador es más fácil de implementar y entender lo que realiza a comparación del segundo operador. La configuración del operador Weight of Information Gain es la siguiente

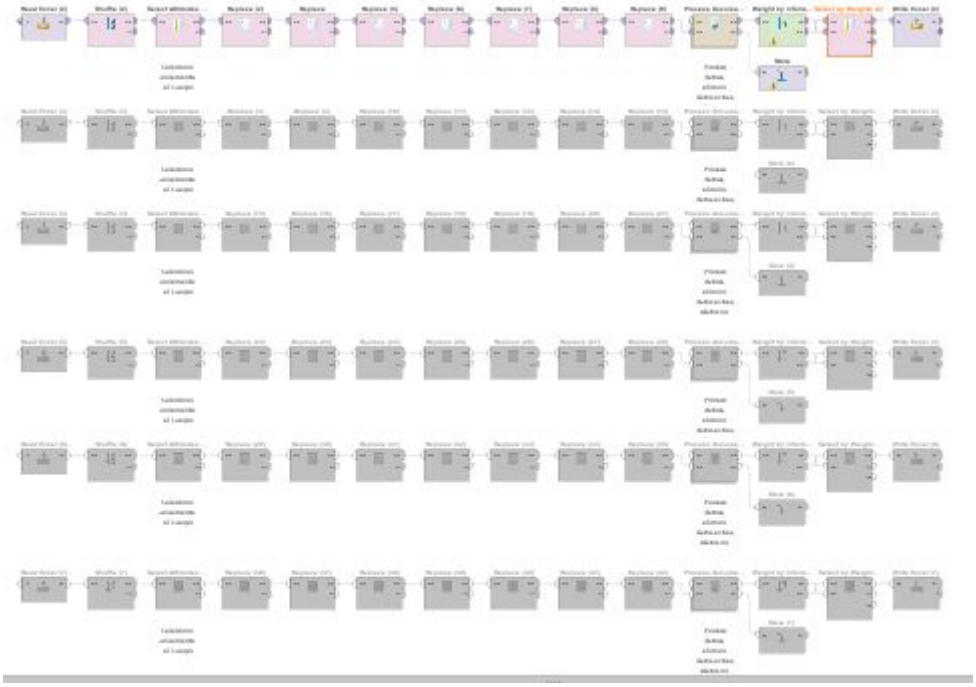


Y el operador **Select by Weight** su configuración de Weight mayor a 0.5

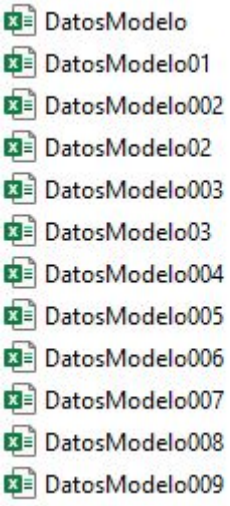



Sin embargo, por razones de estudio para saber las cotas que permitirán mejores precisiones en nuestros modelos entrenado, se procede a intentar con varias cotas (0.002, 0.003, 0.004, 0.005, 0.006, 0.007, 0.008, 0.009). De esta manera se puede hacer un análisis más profundo. Obteniendo así, el siguiente proceso.



La salida de este proceso será un archivo excel, con un total de atributos de 15



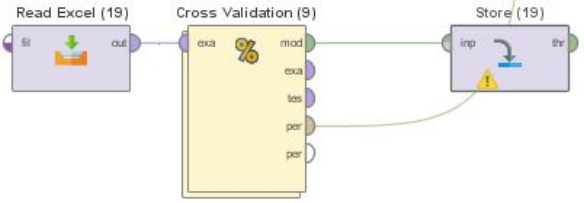

El cual es un vector de documentos con las palabras de los documentos y sus pesos

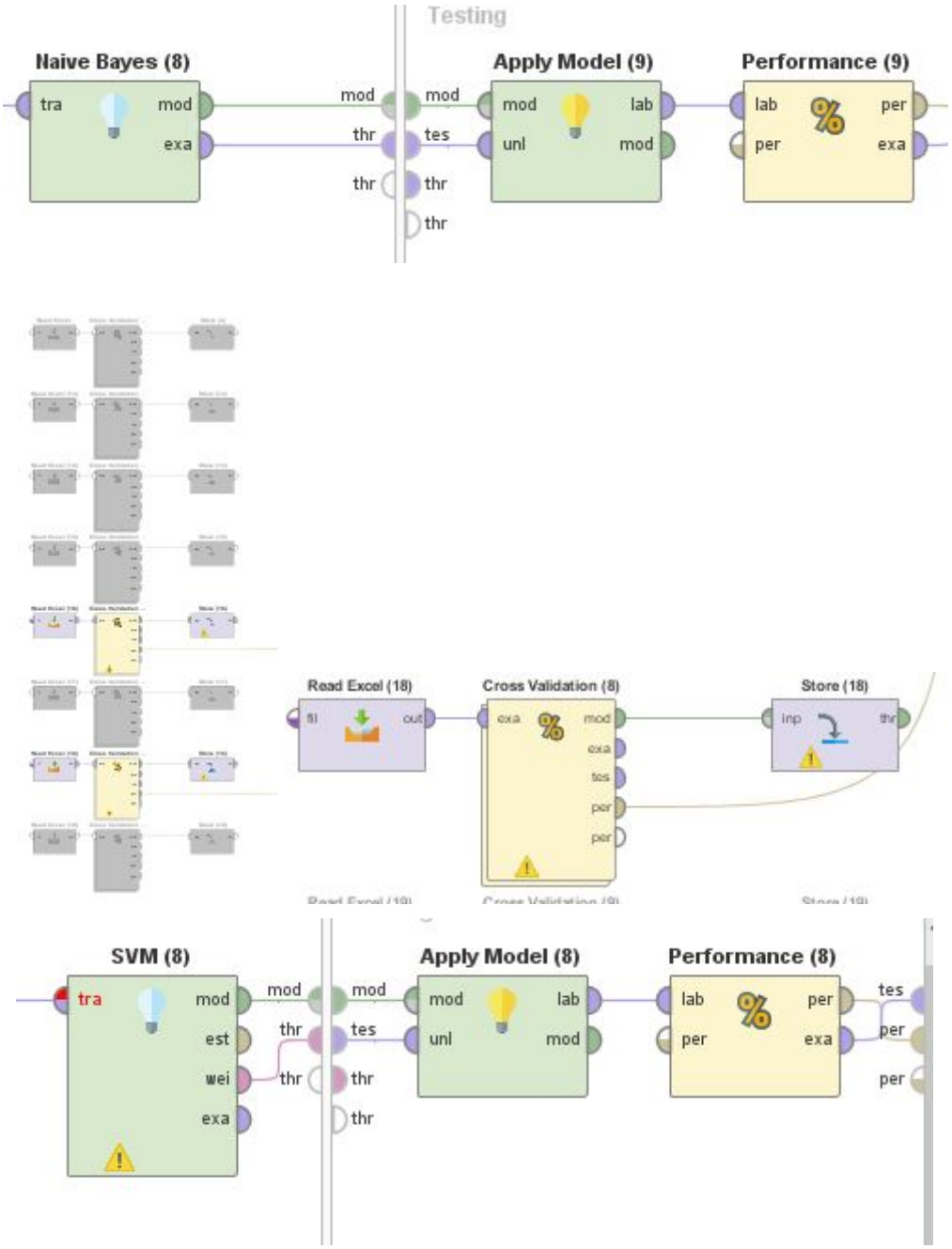
| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | art | captur | check | emot | equal | eventu | flick | funni | predict | scari | similar | stori | superherc | tri | try | Etiqueta |
| 2 | ,0 | ,0 | ,0 | ,0 | ,0 | ,0 | ,2 | ,0 | ,0 | ,0 | ,0 | ,0 | ,0 | ,0 | ,0 | Negativo |
| 3 | ,0 | ,0 | ,0 | ,0 | ,0 | ,0 | ,1 | ,0 | ,0 | ,0 | ,0 | ,0 | ,0 | ,0 | ,0 | Negativo |

**Requisito 4**

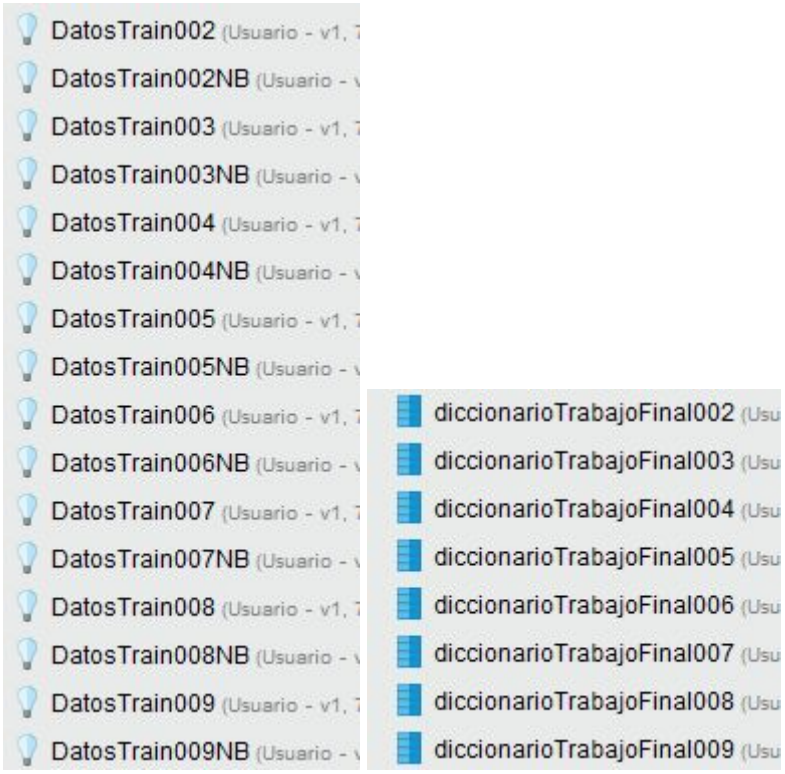
Construir modelo de entrenamiento, obtenido el vector del documento y los pesos de los atributos, se puede experimentar usando diferentes algoritmos de clasificación. Se usa dos algoritmos de entrenamiento Naive Bayes y SVM, para identificar cuál proporciona la mejor precisión además se calcula el rendimiento de los algoritmos de entrenamiento utilizados para la clasificación de las valoraciones de los temas como positivos o negativos.

Archivo : **3-EntrenamientoModeloNB.rmp y 3-EntrenamientoModeloSVM.rmp**






El resultado de estos procesos son los modelos ya entrenados, así como los diccionarios con las palabras y pesos obtenidos




**Entrenamiento del Modelo**

En el caso del uso del algoritmo Naive Bayes, se obtuvieron los siguientes datos:

| Select Weight | True Positive | True Negative | Accuracy | Precision | Clasificación Positiva(de 248) | Clasificación Negativa(de 247) |
|---|---|---|---|---|---|---|
| 0.002 | 58.87 | 54.25 | 56.55 | 56.97 | 146 | 113 |
| 0.003 | 97.58 | 95.95 | 96.76 | 97.61 | 242 | 237 |
| 0.004 | 97.58 | 95.95 | 96.76 | 97.61 | 242 | 237 |
| 0.005 | 95.56 | 93.93 | 94.75 | 95.67 | 237 | 232 |
| 0.006 | 95.56 | 94.33 | 94.96 | 95.70 | 237 | 233 |
| 0.007 | 87.90 | 91.09 | 89.50 | 88.57 | 218 | 225 |
| 0.008 | 88.71 | 95.14 | 91.92 | 89.53 | 220 | 235 |
| 0.009 | 86.69 | 87.45 | 87.09 | 87.10 | 215 | 216 |

En el caso del uso del algoritmo SVM, se obtuvieron los siguientes datos:

| Select Weight | True Positive | True Negative | Accuracy | Precision | Clasificación Positiva(de 248) | Clasificación Negativa(de 247) |
|---|---|---|---|---|---|---|
| 0.002 | 70.97 | 47.37 | 59.18 | 62.19 | 176 | 117 |

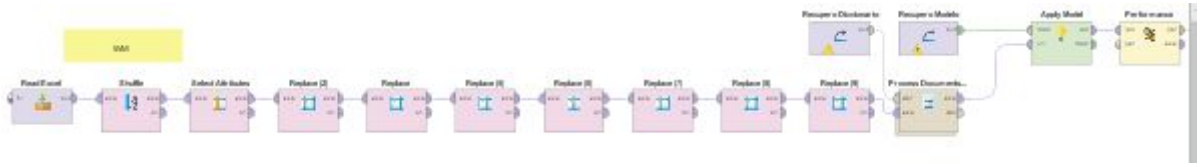
| | | | | | | |
|---|---|---|---|---|---|---|
| 0.003 | 97.18 | 90.69 | 93.94 | 97.15 | 241 | 224 |
| 0.004 | 97.18 | 92.31 | 94.74 | 97.16 | 241 | 228 |
| 0.005 | 95.97 | 89.88 | 92.92 | 95.93 | 238 | 222 |
| 0.006 | 93.95 | 91.5 | 92.73 | 94.03 | 233 | 226 |
| 0.007 | 93.15 | 85.43 | 89.30 | 93.13 | 231 | 211 |
| 0.008 | 82.19 | 93.15 | 87.67 | 92.45 | 231 | 203 |
| 0.009 | 94.35 | 70.85 | 82.66 | 93.04 | 234 | 175 |

**Requisito 5**

**Verificar Modelos**

Aplicar el modelo entrenado. Usando el 50% de los datos originales que fueron separados para tareas de validación es necesario validar el modelo entrenado. Para ello es necesario ejecutar las siguientes actividades: **i) convertir estos datos de prueba en un vector de documento. En otras palabras, tenemos que repetir el proceso del paso 2 (sin los operadores de filtrado y división de datos) en el 50% de los datos que se reservaron para las pruebas, y ii) aplicar el modelo entrenado sobre los datos de prueba y verificar su precisión.**

Archivo **ValidacionModelo.rmp(4-ValidacionModelo.rmp)**

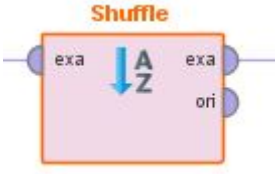


El archivo de ingreso es

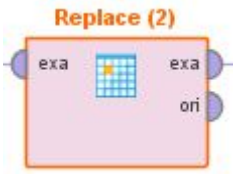
5-Cuerpo-Test

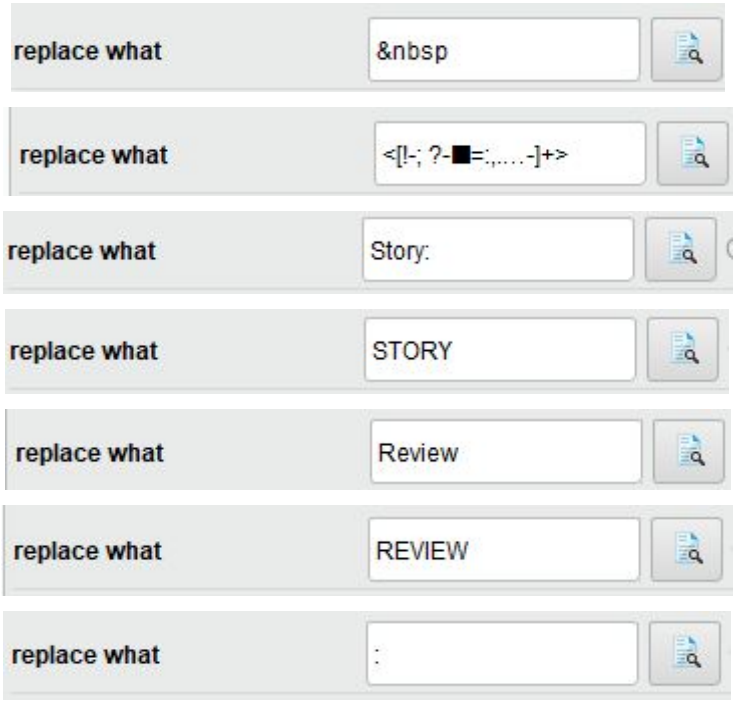
El cual fue el resultado de dividir el corpus en los pasos anteriores, estos datos permitirán validar los modelos.

Se utiliza el operador Shuffle para aleatorizar los datos, así el modelo tendrá una mejor evaluación.

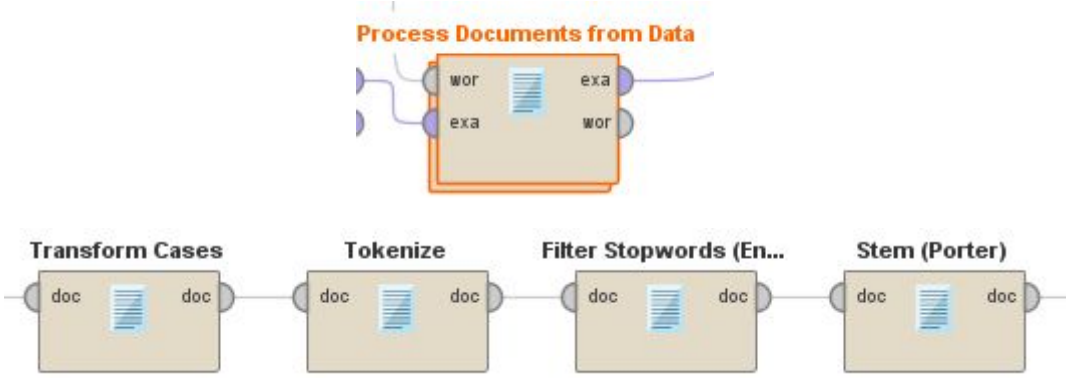



Al igual que los datos de entrenamiento, se debe tratar el texto, por lo cual se utiliza el operador Replace, con distintas configuraciones





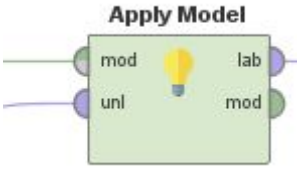

Estos operadores guardan el mismo orden y las mismas expresiones que los puntos de entrenamiento de modelo. Así mismo el operador Process Document from Data, con los subprocesos respectivos.




El procesamiento de datos del corpus de validación se mezclan con los datos previamente de los datos de entrenamiento


Se procede a recuperar el modelo entrenado previamente


Así el resultado del Modelo y Procesamiento de datos se ingresa en el operador




Y finalmente Performance para obtener los datos respectivos a Confusion Matrix, que permite verificar la validez del modelo


Este proceso se repite con todos los casos para poder contrastar los resultados. Los resultados obtenidos se exponen a continuacion:

**Validación de Modelo**

En el caso del uso del algoritmo Naive Bayes, se obtuvieron los siguientes datos:

| Select Weight | True Positive | True Negative | Accuracy | Precision | Clasificación Positiva**(de 248)** | Clasificación Negativa**(de 247)** |
|---|---|---|---|---|---|---|
| 0.002 | 56.68 | 50.81 | 53.74 | 53.74 | 140 | 126 |
| 0.003 | 56.68 | 51.61 | 51.14 | 54.47 | 140 | 128 |
| 0.004 | 56.68 | 51.61 | 54.14 | 54.47 | 140 | 128 |
| 0.005 | 67.78 | 51.21 | 57.98 | 59.35 | 160 | 127 |
| 0.006 | 63.97 | 50.81 | 57.37 | 58.60 | 158 | 126 |
| 0.007 | 58.30 | 53.63 | 55.95 | 56.36 | 144 | 133 |
| 0.008 | 57.09 | 53.63 | 55.35 | 55.65 | 141 | 133 |
| 0.009 | 61.13 | 56.85 | 58.99 | 59.49 | 151 | 141 |

En el caso del uso del algoritmo SVM, se obtuvieron los siguientes datos:

| **Select Weight** | **True Positive** | **True Negative** | **Accuracy** | **Precision** | **Clasificación Positiva (de 248)** | **Clasificación Negativa(de 247)** |
|---|---|---|---|---|---|---|
| 0.002 | 70.04 | 51.21 | 60.61 | 63.18 | 173 | 127 |
| 0.003 | 70.85 | 50.00 | 60.04 | 63.27 | 175 | 124 |
| 0.004 | 70.85 | 50.40 | 60.61 | 63.45 | 175 | 125 |
| 0.005 | 68.42 | 54.44 | 61.41 | 63.38 | 169 | 135 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.006 | 65.59 | 58.47 | 62.02 | 63.04 | 162 | 145 |
| 0.007 | 65.99 | 54.44 | 60.20 | 61.64 | 163 | 135 |
| 0.008 | 69.64 | 46.77 | 58.18 | 60.73 | 172 | 116 |
| 0.009 | 74.90 | 46.77 | 60.81 | 65.17 | 185 | 116 |