CLASIFICACIÓN DE ARTÍCULOS INVESTIGATIVOS SEGÚN EL CONTENIDO DE SUS ABSTRACT

Jairo Castrellón Torres[1]

[1]*Estudiante de Maestría-Estadística, Universidad Nacional de Colombia. Email: jcastrellont@unal.edu.co*

Resumen

Existen muchas plataformas que tienen bases de datos de artículos de investigación y facilitan su busqueda (Scopus, Google académico, Science Direct, etc), en estas plataformas se pueden hacer filtros para encontrar los articulos de interés. Estos filtros se pueden realizar por palabras clave, título del artículo, nombre del autor, entre otros. Sin embargo, cuando el investigador está interesado en un tema en específico, la busqueda por filtros de palabras clave no es suficiente para encontrar cuales son aquellos que realmente sirven para la investigación, lo que puede resultar una tarea tediosa, y no siempre muy útil.

Dada esta situación, este proyecto pretende hacer uso de las herramientas de la minería de datos, teniendo un conjunto de datos que consista de información fundamental (nombre del artículo, palabras clave, abstract, autor, etc) sobre una cantidad considerable de artículos de investigación acerca de un tema en particular como por ejemplo Inteligencia Artificial, se pueda realizar posteriormente una clasificación dependiendo de las palabras usadas en estos artículos, las frases más mencionadas en sus abstract, de tal forma que se puedan hacer agrupaciones de aquellos que hablen de un tema en común, y de esta manera, facilitar más la búsqueda de los temas de interés del investigador.

Keywords: Clasificación de textos, minería de textos, Agrupación de datos textuales

1 Introducción

El presente trabajo muestra el preprocesamiento y análisis de un conjunto de datos que contiene información básica, como el título, abstract, autor, etc, de 41000 artículos de investigación. Aunque por costo computacional, no se trabajo con el total de los datos, si se hizo una selección aleatoria de 2000 de ellos para hacer el análisis.

En la sección 2, se hablará del conjunto de datos y de los objetivos de este proyecto, en la sección 3 se mostrará el correspondiente preprocesamiento de los datos, en la sección 4 se dará una descripción de las reglas de asociación encontradas en los datos, en la sección 5 algunos de los resultados de la agrupación, en la sección 6 la clasificación de los textos y por último, algunas conclusiones generales.

2 Descripción de los datos

La idea principal en la búsqueda de los datos es encontrar aquellos que brinden la información necesaria y suficiente con la que se pueda categorizar un artículo, lo deseable sería contar con un conjunto de datos que contenga los artículos completos para hacer una clasificación más exitosa, sin embargo, la obtención de este tipo de datos tiene grandes dificultades, tanto para la obtención como para la manipulación, es por esto que se cuenta únicamente con la información más relevante de los artículos investigativos.

El conjunto de datos con el que se trabaja contiene información relacionada a 41000 artículos de investigación en Machine Learning, Inteligencia Artificial, entre otros, publicados entre 1992 y 2018. Este dataset contiene 9 atributos: autor, dia, mes, año de publicación, abstract, título, código del

(2020)

Universidad Nacional de Colombia

Edited by

artículo y tag. Este dataset está disponible en https://www.kaggle.com/neelshah18/arxivdataset en formato json.

De este podemos observar que los datos se presentan con algunos inconvenientes en su formato y visualización, por lo que es necesario algunas transformaciones para su mejor entendimiento.

3 Pre-procesamiento y análisis exploratorio de datos

Una vez recolectados los datos con los que se trabaja, es necesario un análisis de éstos, para saber información a priori que estos nos puedan brindar, para ello, es necesario hacer pre-procesamiento de los datos, además por que con base en esto es que se desarrollará todo el trabajo de agrupación y clasificación.

Inicialmente, como solo se quiere hacer uso de los abstract para realizar la clasificación, se filtran estos datos de todas las observaciones con las que se cuenta. En este caso, debido a la gran cantidad de datos con los que se cuenta, se seleccionaron 2000 observaciones escogidas aleatoriamente para tener un mejor rendimiento computacional. De estos 2000 datos (abstract) seleccionados, se observó que no se cuentan con datos vacíos ni aparente ruido en ellos, pues es de esperarse que al escribir un artículo de investicació, estos esten bien escritos y cuenten con buena ortografía, por lo que no hubo que hacer arreglos de este tipo.

Por otro lado, para el análisis que se quiere realizar, si es necesario eliminar los datos que no aporten información o que no permitan evidenciar carácteristicas que puedan tener estos datos. Para esto, se realizo este pre-procesamiento en tres etapas: eliminación de información no influyente, tokenizing, y lematizing o stemmizing, que se describe a continuación:

3.1 Eliminación de información no influyente

Como se mencionaba anteriormente, hay datos, en este caso palabras, que son muy frecuentes pero que no aportan mucho a la investigación. Un ejemplo de esto son las denominadas “Stop-words”, que consisten de aquellas palabras que funcionan como conectores, o como artículos, en este caso, al ser textos de investigación escritos en inglés, ejemplos de estas palabras son: *The, of, in, on, as,* etc. Por lo tanto, una primera tarea, es eliminar estas palabras de los abstract a trabajar.

Otra tarea, es eliminar los números de estos textos, pues tampoco brindarán mucha ayuda a la hora de hacer clasificación. Por otro lado, hay otras palabras, que también son muy comunes pero al contrario de las stopwords, si proporcionan información pero que en este caso no es útil, es decir, en este caso, los datos recolectados, se saben en un primer momento que son de artículos investigativos que tratan de temas relacionados con inteligencia artificial, por lo tanto, todas estas palabras relacionadas, también fueron eliminadas del diccionario a trabajer, como por ejemplo *Artificial, Machine, Learning, Big, Data, Models,* etc.

Por último, se eliminaron aquellas palabras que tenían muy baja frecuencia en el conjunto de datos, puesto que al ser poco mencionadas, con seguridad no iban a ser importantes a la hora de clasificar los textos.

3.2 Tokenizing

El proceso de tokenización consiste en que las palabras con las que se cuenta, puedan contraerse en una base, reduciendo así el número de palabras que significan en esencia lo mismo, o provienen del mismo origen, por ejemplo: learn, learning, learned, pueden todas asociarse a la misma base learn. En este caso, se realizó una vectorización de los abstract disponibles y posteriormente se hizo la correspondiente tokenización.

3.3 Lematizing o stemmizing

En esta etapa, no es que los dos procesos sean iguales, solo que se permite hacer uso del uno o del otro según se considere más adecuado. Para este proyecto se realizaron los dos procesos de tal manera que se pudiera tener información más concreta. Estos procesos consisten en dejar cada palabra en una base aún mas simplificada de lo que puede hacerse en la anterior etapa, de tal manera, que toda palabra que siquiera se parezca, sea agrupada dentro del mismo significado, por ejemplo, siguiendo con el caso antes mencionado, se puede reducir learn a lern.

Algunos ejemplos de como al final quedan estos datos, se muestran a continuación:

```
0       exist method domain adapt da work assumpt taskrelev targetdomain
train data given howev assumpt violat often ignor prior work tackl issu
propos zeroshot deep domain adapt zdda use privileg inform ...
1       season distinct characterist often observ mani practic time seri
artifici neural network ann class promis model effici recogn forecast season
pattern paper particl swarm optim pso approach use enh...
2       consid problem recov blockspars signal whose structur unknown empha
priori blockspars signal nonzero coeffici occur cluster aris natur mani
practic scenario howev knowledg block structur usual una...
```

Un análisis adicional a los antes mencionados, se pudo realizar al obtener algunas otras transformaciones. Por ejemplo, se realizaron todas las posibles combinaciones de dos palabras y de tres palabras, o lo que se denominan como **bigramas** y **trigramas**, obteniendo que las más comunes, respectivamente son:

```
convolut neural
deep neural
loss function
natur languag
neural cnn
object detect
recurr neural
```

```
convolut neural cnn
deep convolut neural
markov decis process
natur languag process
support vector machin
```

Lo que permite hacerse a un análisis de como se comportan los artículos con los que se esta trabajando y los temas puntuales que estos tratan.

4 Asociación

Se aplicó asociación al conjunto de datos. Para esto, se usaron los algorítmos apriori y FPgrowth para comparar los resultados, de tal manera que se pueda especificar cuál de estos nos dá mejores resultados.

4.1 Algoritmo apriori

Para el algorítmo apriori, se modificaron los datos de tal manera que se obtuviera una matríz con cada uno de los núcleos de las palabras obtenidas en el proceso de lemmatization y stemming.

En este punto, fue necesario eliminar filas (abstracts) de la tabla debido a la longitud de su abstract, algunas con muchas palabras y otras con muy pocas, todo esto para no generar ruido a la hora de realizar la asociación.

Al hacer uso del algorítmo de asociacón apriori, con un soporte mínimo de 4 % y una confianza del 70 % se obtuvieron las siguientes reglas:

```
Rule: deep -> neural
Support: 0.0968586387434555
Confidence: 0.9024390243902439
Lift: 4.996111700247438
====================================
Rule: neural -> network
Support: 0.06282722513089005
Confidence: 1.0
Lift: 6.821428571428572
====================================
```

Lo que ratifica lo que se veía anteriormente cuando se explicaba la frecuencia de las palabras y la temática de los artículos escogidos. Es decir, esto nos confirma que la temática principal en la mayoría de artículos es acerca de algorítmos de aprendizaje de máquina pues alcanzando el soporte mínimo, cada vez que apareció la palabra deep tambien apareció neural, de igual manera con las palabras neural y network, haciendo referencia a la relevancia de el tema de redes neuronales en los artículos trabajados.

Ahora, modificando el soporte, llevandolo al 2 % y aumentando la confianza al 90 % se obtienen diferentes reglas, aparte de las que ya obtuvimos anteriormente, como las siguientes:

```
Rule: method -> machin
Support: 0.020942408376963352
Confidence: 1.0
Lift: 5.536231884057971
====================================
Rule: neural -> paper
Support: 0.02356020942408377
Confidence: 1.0
Lift: 6.821428571428572
====================================
Rule: use -> neural
Support: 0.02617801047120419
Confidence: 1.0
Lift: 6.821428571428572
====================================
```

explicando la importancia de las palabras aquí mencionadas. Por último, aumentando el soporte mínimo al % y disminuyendo la confianza al % se obtiene:

```
Rule: machin -> learn
Support: 0.0968586387434555
Confidence: 0.5362318840579711
Lift: 4.996111700247438
====================================
```

Lo que confirma de nuevo la relevancia de los temas relacionados a machine learning en los artículos seleccionados.

4.2 Algorítmo FPgrowth

En este caso se hace uso de la tabla con los datos como se muestran en la Figura 8, se establece un soporte mínimo del 30 % y una confianza del 90 % y se obtiene lo siguiente:

```
 support itemsets
0 0.746073 (use)
1 0.617801 (paper)
2 0.536649 (result)
3 0.526178 (data)
4 0.505236 (propos)
5 0.494764 (learn)
6 0.421466 (model)
7 0.421466 (algorithm)
8 0.408377 (method)
9 0.397906 (base)
10 0.358639 (show)
11 0.340314 (approach)
12 0.400524 (perform)
13 0.471204 (use, paper)
14 0.416230 (use, result)
15 0.332461 (result, paper)
16 0.410995 (machin, learn)
17 0.319372 (paper, data)
```

Se puede observar que FPgrowth, aunque saca más reglas, éstas no tinene la misma presición que las obtenidas con el algorítmo apriori. Nuevamente aparece la regla machin learn, lo que nuevamente refleja la importancia del tema en estos artículos.

5 Agrupación

Para realizar agrupación en el conjunto de datos con el que se está trabajando, se realizó mediante dos algoritmos, uno de ellos jerárquico y el otro particional, que en este caso fue el K-means, a continuación se describe la metodología para ellos, sin embargo, la importancia de estos algoritmos está en la definición de las medidas de próximidad y similitud.

En este caso, para ambos algoritmos, se utilizó la medida **Tf-idf** que asigna un peso a cada palabra dependiendo de su influencia sobre todo el documento, de esta manera, una palabra con una medida Tf-idf cercana a 1, indicará que es bastante influyente en su documento, y su medida sera cercana a 0 si no lo es.

Con base en lo anterior, se uso la similaridad coseno, para establecer que tan próximos, en términos de similitud, están los abstracts seleccionados.

5.1 Dendograma

Se realizó el dendograma correspondiente al algorítmo jerárquico, aunque debido a la cantidad de datos, no se puede hacer una identificación clara de la agrupación, si generó algunas etiquetas para los clusters.

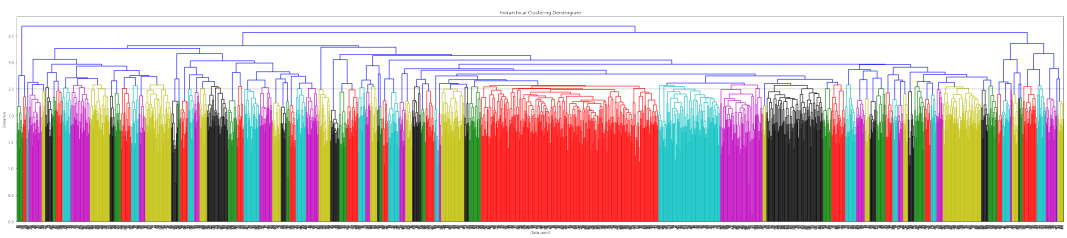

Figura 1. Dendograma de enlace completo producido por el algorítmo de agrupación jerárquico

Generando 289 grupos diferentes, de tal manera que etiquetó a los abstract de esta manera. algunas frecuencias, con los grupos más importantes fueron:

label	Freq
166.0	75
191.0	48
170.0	27
160.0	25
175.0	23

5.2 K-MEANS

Haciendo uso de este algorítmo, usando las medidas de proximidad y similitud mencionadas anteriormente, se obtuvo una agrupación más acorde a lo que se busca en este tipo de problemas. Inicialmente, se realizo la gráfica de codo (Elbow) de tal manera que pudiera dar indicios de la cantidad de clusters necesarios para la agrupación. La curva obtenida se puede ver en la figura 2

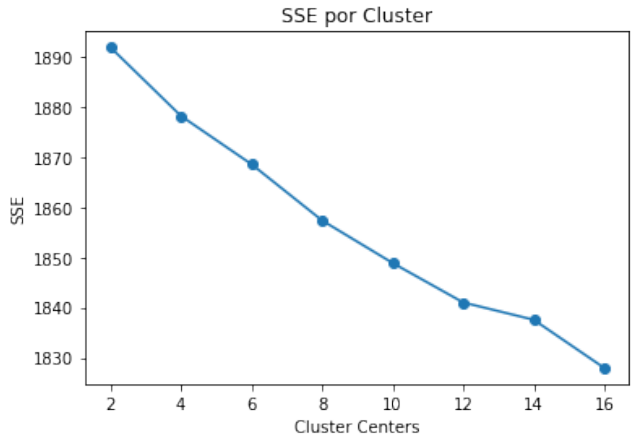

Figura 2. Gráfica Elbow para el número de clusters vs la suma de cuadrados del error dentro de clusters

De donde se concluyó que para una mejor agrupación, eran convenientes un total de 16 clusters, los cuales se obtuvieron, teniendo en cuenta las palabras que los abstract tenían, como se muestra en el cuadro 1

Cuadro 1. Clusters obtenidos por el algorítmo K-Means según las palabras contenidas

Cluster	Words
0	estim, optim, approxim, factor, number, point, subspac, kmean, matrix, cluster
1	reason, theori, search, solv, optim, fuzzi, rule, constraint, program, logic
2	neural, experi, accuraci, represent, metric, onlin, evalu, recommend, user, predict
3	express, video, appear, attribut, object, tracker, recognit, facial, track, face
4	point, applic, scene, reconstruct, camera, depth, motion, pose, shape, estim
5	latent, relat, classifi, embed, manifold, topic, space, label, classif, class
6	minim, polici, regret, convex, function, bound, stochast, gradient, optim, converg
7	transfer, discrimin, classifi, sourc, target, adapt, attack, gan, domain, adversari
8	queri, process, nois, probabl, distribut, quantum, approxim, answer, sampl, question
9	classif, detect, recognit, object, architectur, convolut, layer, neural, deep, cnn
10	transcript, asr, phonem, neural, corpu, acoust, recognit, languag, speaker, speech
11	interact, scenario, adapt, decis, navig, reinforc, knowledg, environ, agent, robot
12	human, process, machin, energi, research, develop, plan, ai, control, intellig
13	sentiment, inform, annot, region, extract, object, video, text, detect, segment
14	condit, measur, select, causal, bayesian, kernel, infer, graph, structur, variabl
15	vector, corpora, translat, represent, text, english, embed, semant, languag, word

Se puede notar que, sin hacer un análisis exhaustivo de cada artículo, que por ejemplo, aquellos abstract que quedaron agrupados en el cluster 13, hablan de temas relacionados al análisisde textos, así como los que están agrupados en la cluster 10 hablan de reconocimiento de voz o los del cluster 9 hablan de clasificación usando modelos de deep learning.

6 Clasificación

Para la clasificación de los textos, se tomó como referencia los clusters asignados a cada abstract de cada artículo, y se probaron los diferentes modelos de clasificación, obteniendo como los de mejor rendimiento según su accuracy, los modelos de Multinomial Naive Bayes y redes neuronales.

En ambos casos, la modelación se realizo haciendo uso de la validación cruzada para mitigar el sobreajuste de los modelos y lograr una mejor clasificación de los datos.

6.1 Multinomial Naive Bayes

Un algorítmo muy útil, además de muy usado para la clasificación de textos es el multinomial Naive Bayes, debido a que la distribución probabilistica multinomial permite describir la proporción de palabras que son mas influyentes dentro del documento, tal como lo realiza la medida Tf-idf.

Al ajustar el modelo a nuestrus datos, dividiendo el conjunto de datos en 30 partes para hacer validación cruzada, se obtuvo un accuracy de:

```
The accuracy of the model is: 0.7138377311183043
```

Aunque este no es el mejor resultado ni el que se esperaba, hay que decir que con respecto a otros, se ha comportado mucho mejor. En el cuadro 2 se ven los indicadores del clasificador por cada etiqueta.

Cuadro 2. Indicadores por etiqueta del clasificador Multinomial Naive Bayes

label	precision	recall	f1-score	support
0.0	0.78	0.55	0.65	65
1.0	0.82	0.82	0.82	204
2.0	0.71	0.58	0.64	283
3.0	0.70	0.75	0.73	65
4.0	0.75	0.49	0.59	55
5.0	0.70	0.72	0.71	127
6.0	0.68	0.90	0.77	140
7.0	0.83	0.65	0.73	62
8.0	0.76	0.49	0.59	107
9.0	0.68	0.83	0.75	209
10.0	0.88	0.70	0.78	40
11.0	0.71	0.71	0.71	77
12.0	0.78	0.68	0.72	105
13.0	0.68	0.69	0.68	142
14.0	0.67	0.79	0.73	213
15.0	0.60	0.81	0.69	75

Mientras que en la figura 3 se muestra la matriz de confusión del clasificador.

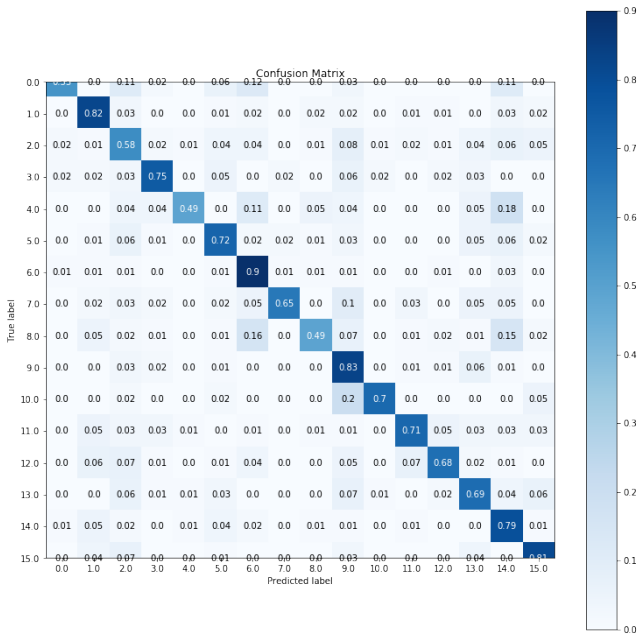

Figura 3. Matríz de confusión para el clasificador Multinomial Naive Bayes

6.2 Neural Network

En este caso, de nuevo utilizando las etiquetas proporcionadas por la agrupación realizada anteriormente, se ajusto una red neural de una capa oculta con 500 nodos, de tal manera que permitiera realizar una clasificación más efectiva. Nuevamente, se utilizó validación cruzada para

mitigar el sobreajuste y no memorizar el error en la clasificación. Se obtuvo como resultado que

```
The accuracy of the model is: 0.8293098170076159
```

Obteniendo un rendimiento mucho mejor que el clasificador anterior. En el cuadro 3 se ven los indicadores del clasificador por cada etiqueta.

Cuadro 3. Indicadores por etiqueta del clasificador Neural Network

label	precision	recall	f1-score	support
0.0	0.84	0.83	0.84	65
1.0	0.87	0.87	0.87	204
2.0	0.76	0.77	0.77	283
3.0	0.84	0.83	0.84	65
4.0	0.85	0.91	0.88	55
5.0	0.81	0.85	0.83	127
6.0	0.82	0.89	0.86	140
7.0	0.80	0.79	0.80	62
8.0	0.81	0.65	0.73	107
9.0	0.87	0.89	0.88	209
10.0	0.86	0.93	0.89	40
11.0	0.73	0.77	0.75	77
12.0	0.77	0.70	0.74	105
13.0	0.90	0.93	0.91	142
14.0	0.87	0.79	0.83	213
15.0	0.84	0.91	0.87	75

Mientras que en la figura 4 se muestra la matriz de confusión del clasificador.

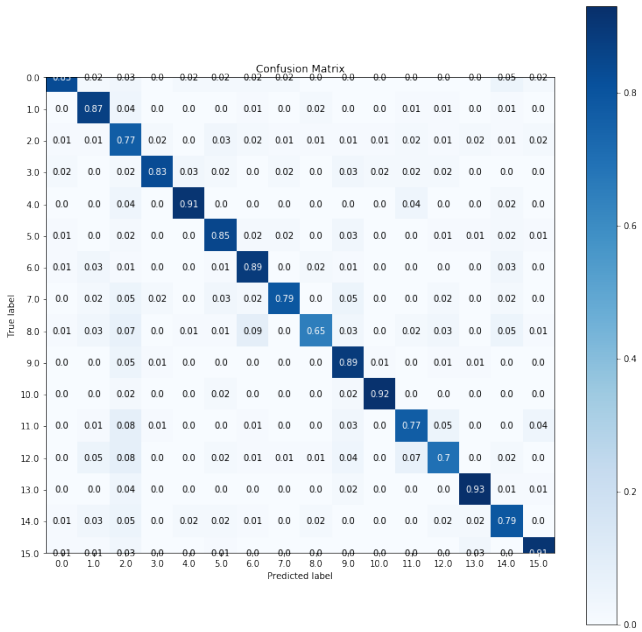

Figura 4. Matríz de confusión para el clasificador Neural Network

7 Conclusiones

Entre algunas conclusiones importantes de este trabajo son la importancia del buen pre procesamiento de los datos, es de vital importancia hacer una limpieza adecuada de los datos, junto con la necesidad de adecuarlos de tal manera que los clasificadores o algorítmos que se vayan a usar, sean efectivamente útiles.

También, se puede concluir que en el trabajo con datos textuales es importante discretizar la información que se tiene, obteniendo más datos de los que el dataset nos brinda, de esta manera se tienen más herramientas para trabajar.

Otra conclusión es la busqueda adecuada de los clasificadores, en este caso, los dos encontrados se comportaron de muy buena manera, pero también se vió que muchos de los tradicionales como los árboles de decisión, random forest, entre otros, no son buenos para los datos textuales. De esta misma manera ocurre en la agrupación, no todos los algorítmos son aplicables a este tipo de datos, por lo que es necesario una buena manipulación de ellos.